LIFE OF A pixel

Steve Kobes
skobes@chromium.org

June 2018

slides: bit.ly/lifeofapixel

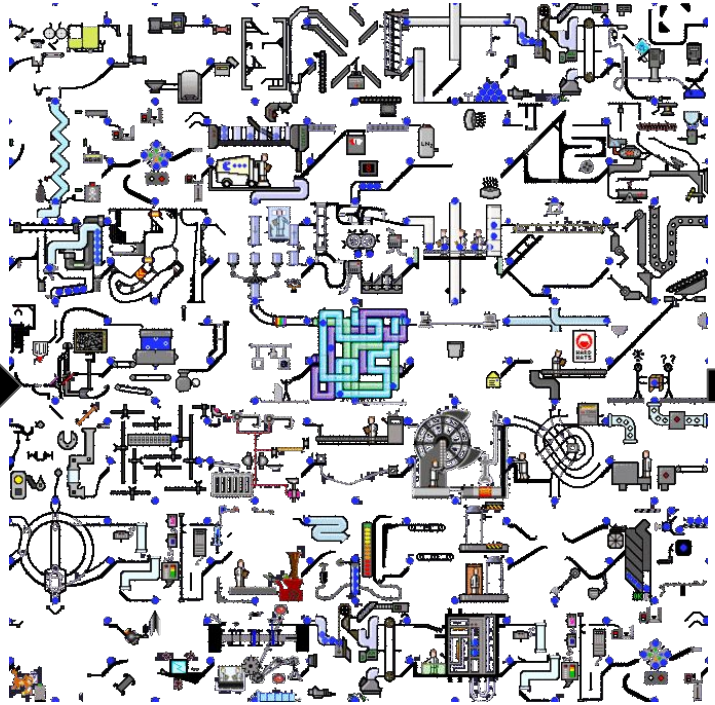
"The unexamined pixel is not worth rendering."
— Socrates

LIFE OF A pixel

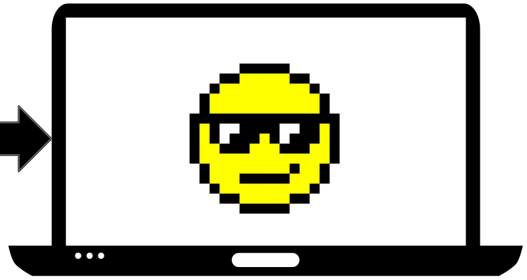
web content

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="u
    <title>Page name<
    <meta name="descri
    <link href="assets/
    <link rel="shortcut
  </head>
```

magic



pixels



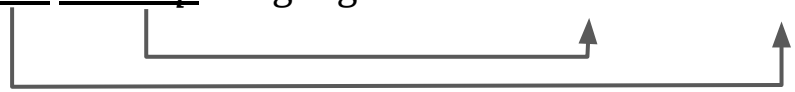
LIFE OF A pixel

68 69.0.3445.0 70

June 2018

(web) content

- **HTML** = *Hyper-Text Markup Language*



```
<p> hello </p>
```

- **CSS** = *Cascading Style Sheets*

```
p { color: red }
```

selector *property*

- **JS** = *JavaScript*

```
p.innerHTML = "goodbye";
```

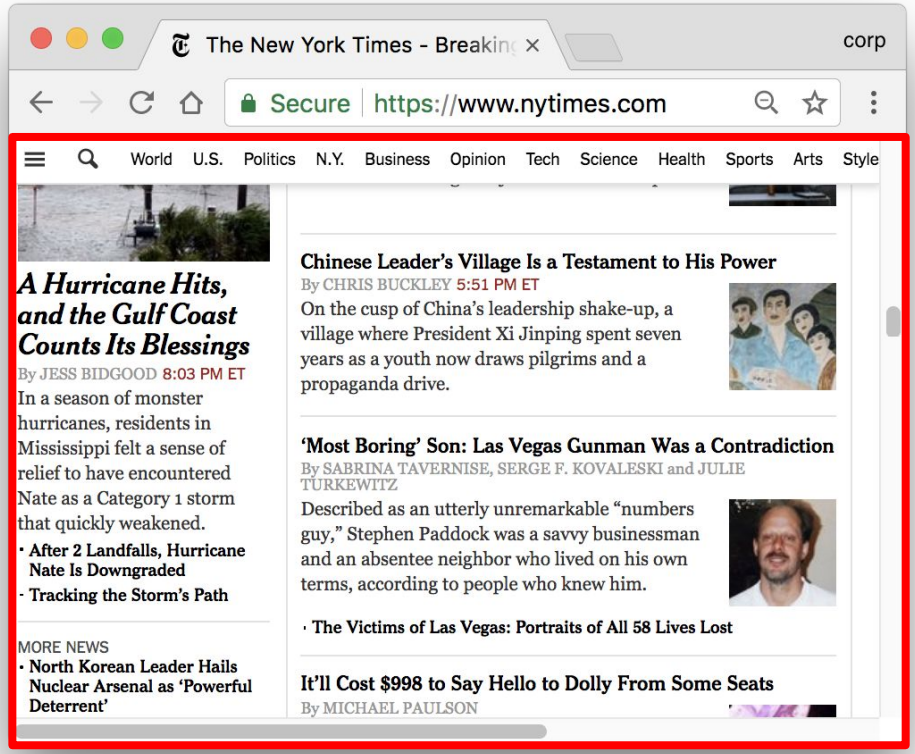
(also images, video, WebAssembly, ...)

content

https://www.nytimes.com

```
    </div>
</div>
            <div class="column">
                <div class="collection crosswords-collection">
                    <article class="story">
                        <h3 class="kicker">
                          <a href="http://wordplay.blogs.nytimes.com">Wordplay
                        </h3>
                                                </article>
                </div>
            </div><!-- close column -->
        </div><!-- close layout -->
    </div><!-- close crossword-subscription -->
</div><!--close collection -->

        </section><!-- close user-subscriptions -->

        <div id="HPMiddle" class="ad hpmiddle-ad nocontent robots-nocontent"></

        <div class="region c-column-middle-span-region">

            <div class="collection">
            <link rel="stylesheet" type="text/css" href="https://int.nyt.com/ap

<style type="text/css">
/*HIDE WATCHING HEADER*/
.portal-container>header { display: none }
</style>

<div id="nytint-hp-watching">
  <div class="portal-container">
    <header class="portal-header">
      <h4>Watching</h4>
    </header>
    <div class="portal-posts-frame expanded"></div>
    <footer class="portal-footer"></footer>
  </div>
</div>

<script type="text/javascript">
require(['foundation/main'], function() {
  require(['homepage/main', 'https://int.nyt.com/apps/portals/assets/portal/app
], function() {
    require(['portal/app'], function(Portal) {

      var opts = {
        env: 'production_published',
        matchHeight: {
          match: '.span-ab-layout.layout > .ab-column',
          container: '.c-column.column',
          maxHeight: 2000
        }
      }

      if (window.location.search.indexOf('portal_variant=watchingNoScroll') !==
        opts.variation = 'simple';
        opts.poll = false;
        opts.limit = 20;
      }

      var watching = Portal.create('#nytint-hp-watching', opts);
```

content



NOT "content"

`content::WebContents`

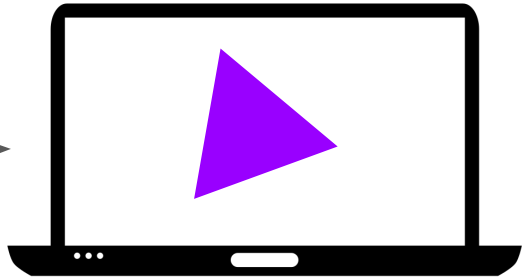
sandboxed renderer process

Blink

pixels

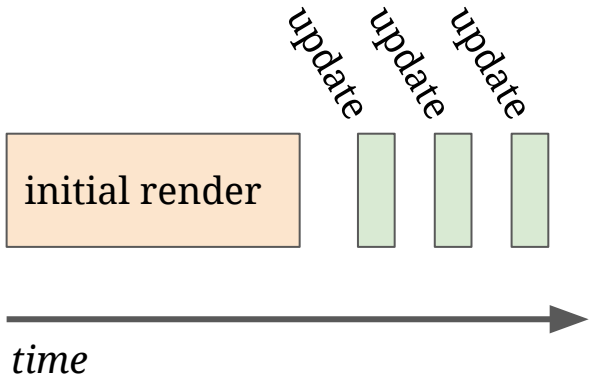
```
#include <GL/gl.h>
```

- textures
- shaders
- vertex buffers
- …



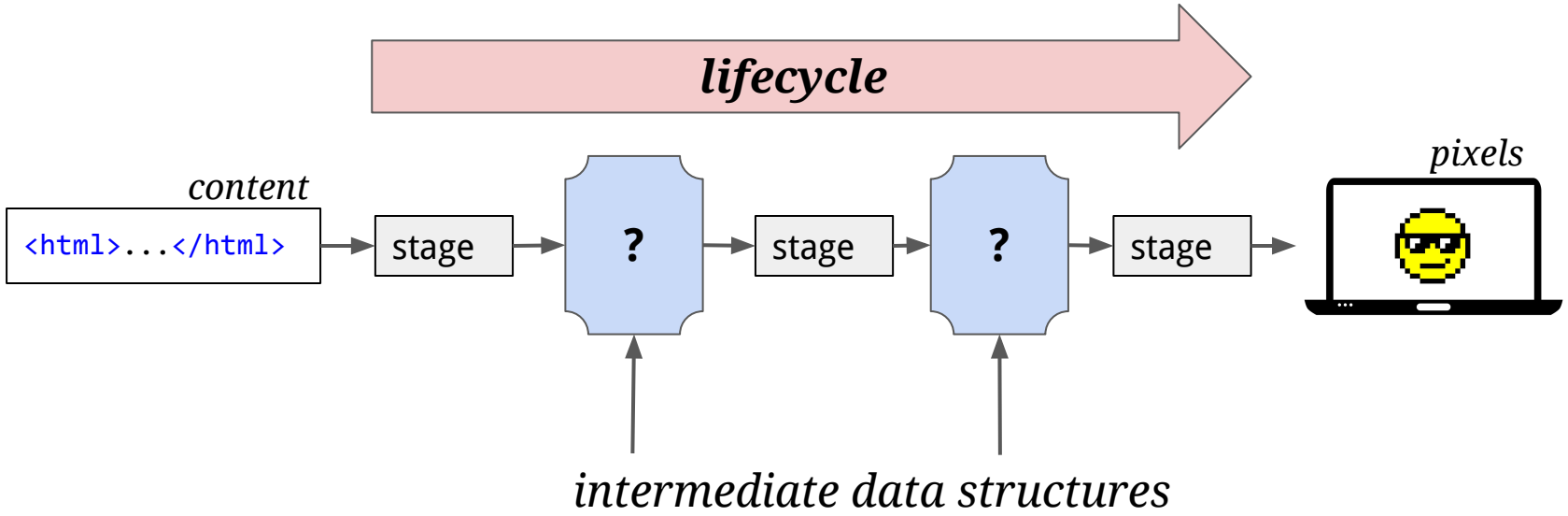
goals

1. *render content into pixels*
2. *build data structures to enable **updating** the rendering efficiently*



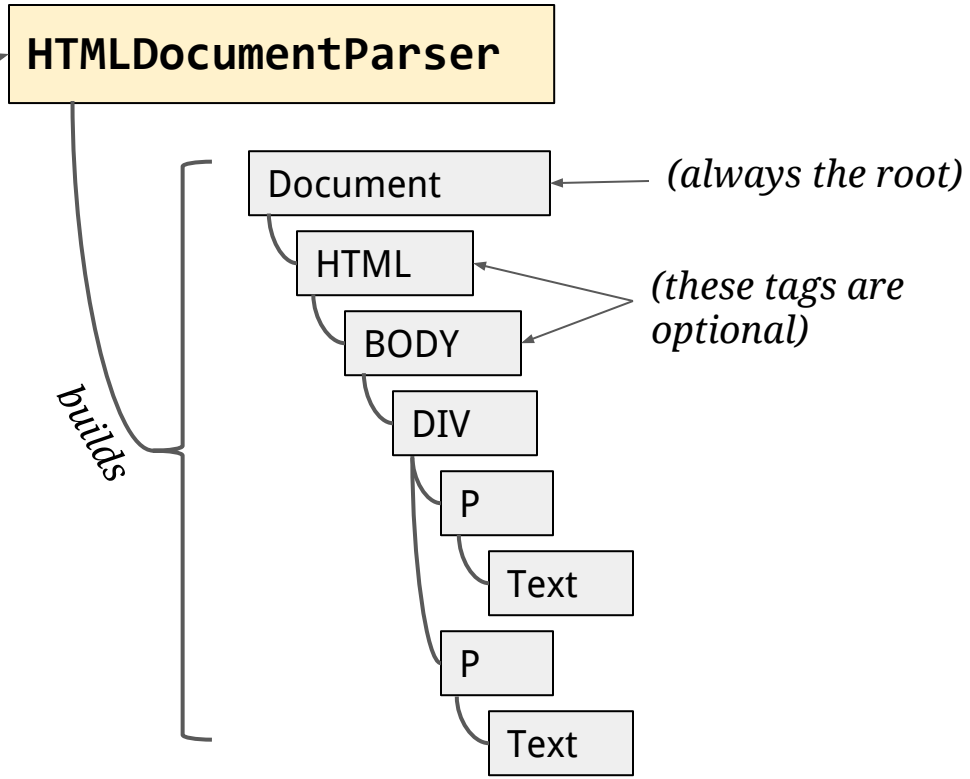
- JavaScript
- user input
- asynchronous loading
- animations
- scrolling
- zooming

stages


lifecycle
content
<html>...</html>
stage
?
stage
?
stage
pixels
intermediate data structures

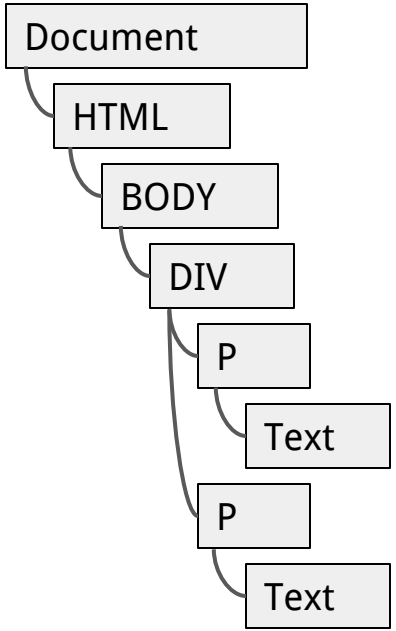
parsing

```
<div>
  <p> hello </p>
  <p> world </p>
</div>
```



- *HTML tags can nest*
- *an **object model** reflects the containment relationships*

DOM

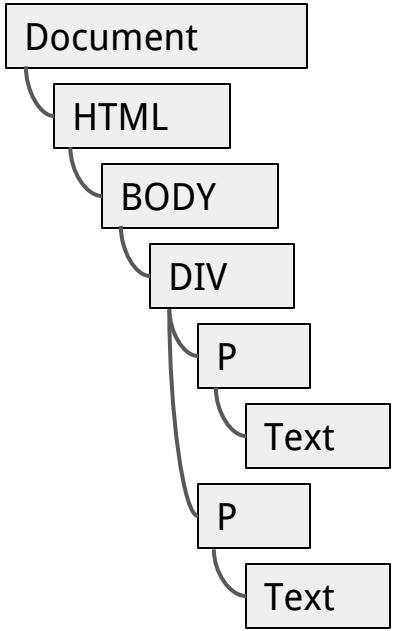


This is the
***D**ocument*
***O**bject*
***M**odel.*

The DOM is a tree!

- *parents*
- *children*
- *siblings*

DOM

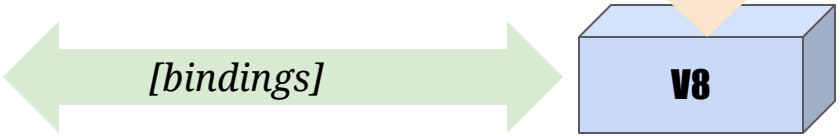


The DOM is both

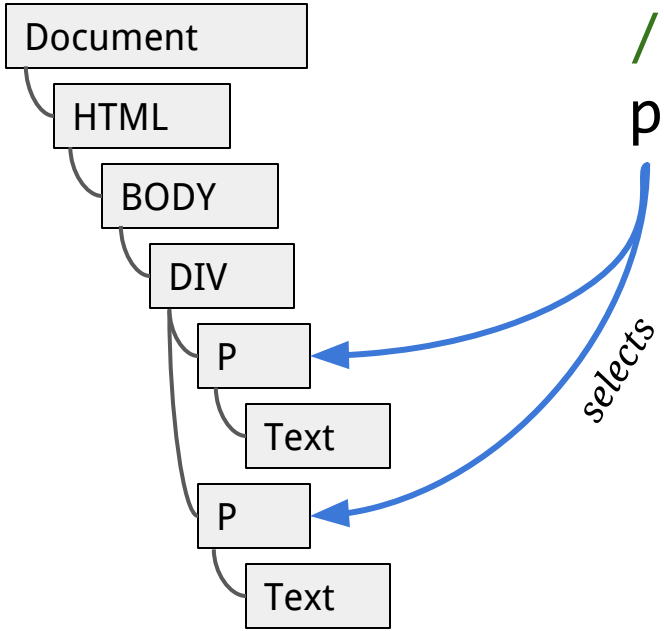
- *Chrome's internal representation*
  AND
- *the API exposed to JavaScript*

```
var div = document.body.firstChild;
var p2 = div.childNodes[1];
p2.appendChild(document.createElement("span"));
p2.foo = {arbitrary: "state"};
```

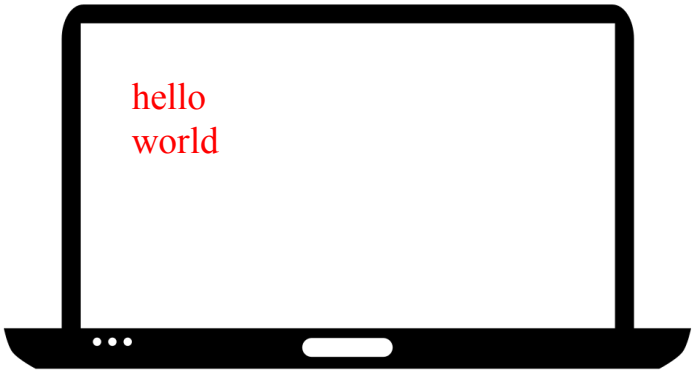
[JS]



style




```
/* every <p> has red text */
p { color: red }
```



style

```
font-weight: bold;
```

hello

```
margin-left: 2cm;
```

hello

```
outline: dashed blue;
```

hello

```
transform: rotate(20deg);
```

hello

```
background: url(kitten.jpg);
```

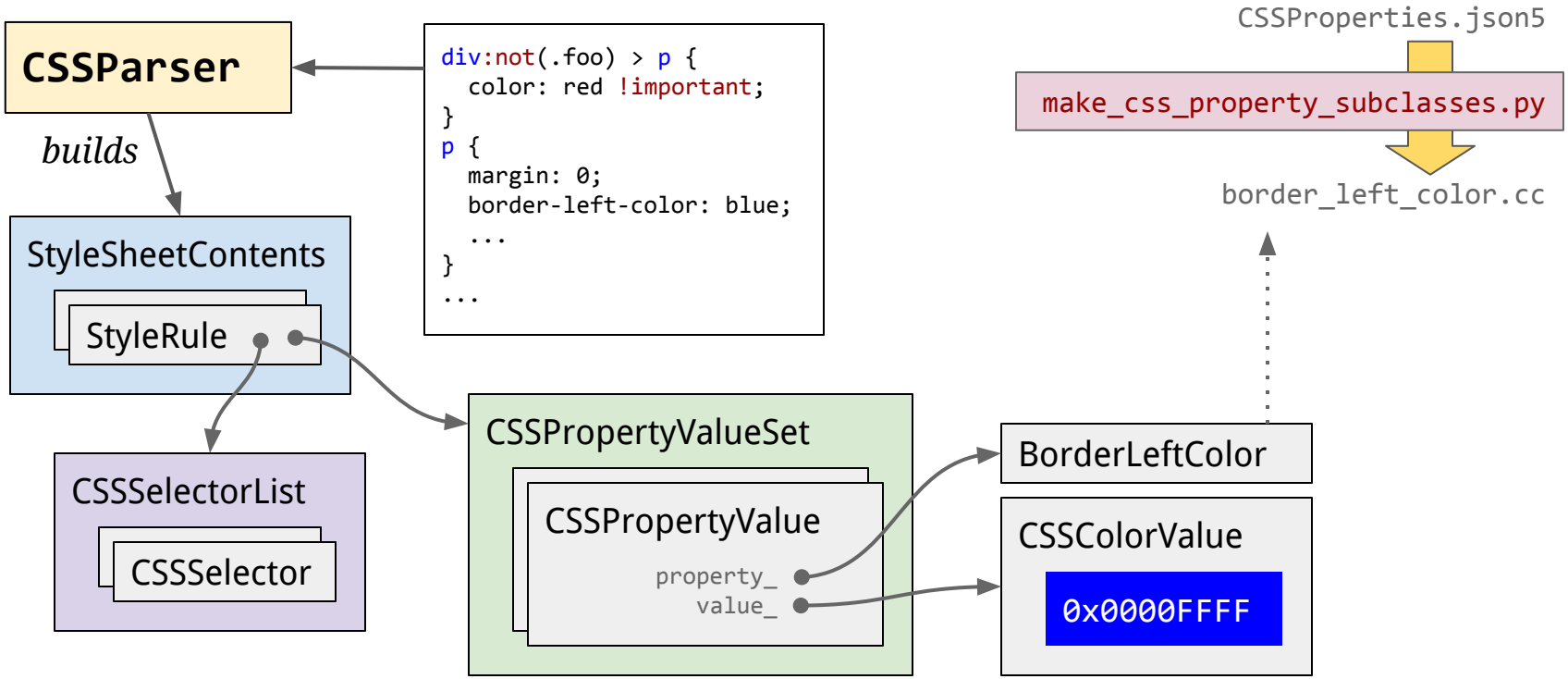
hello

style

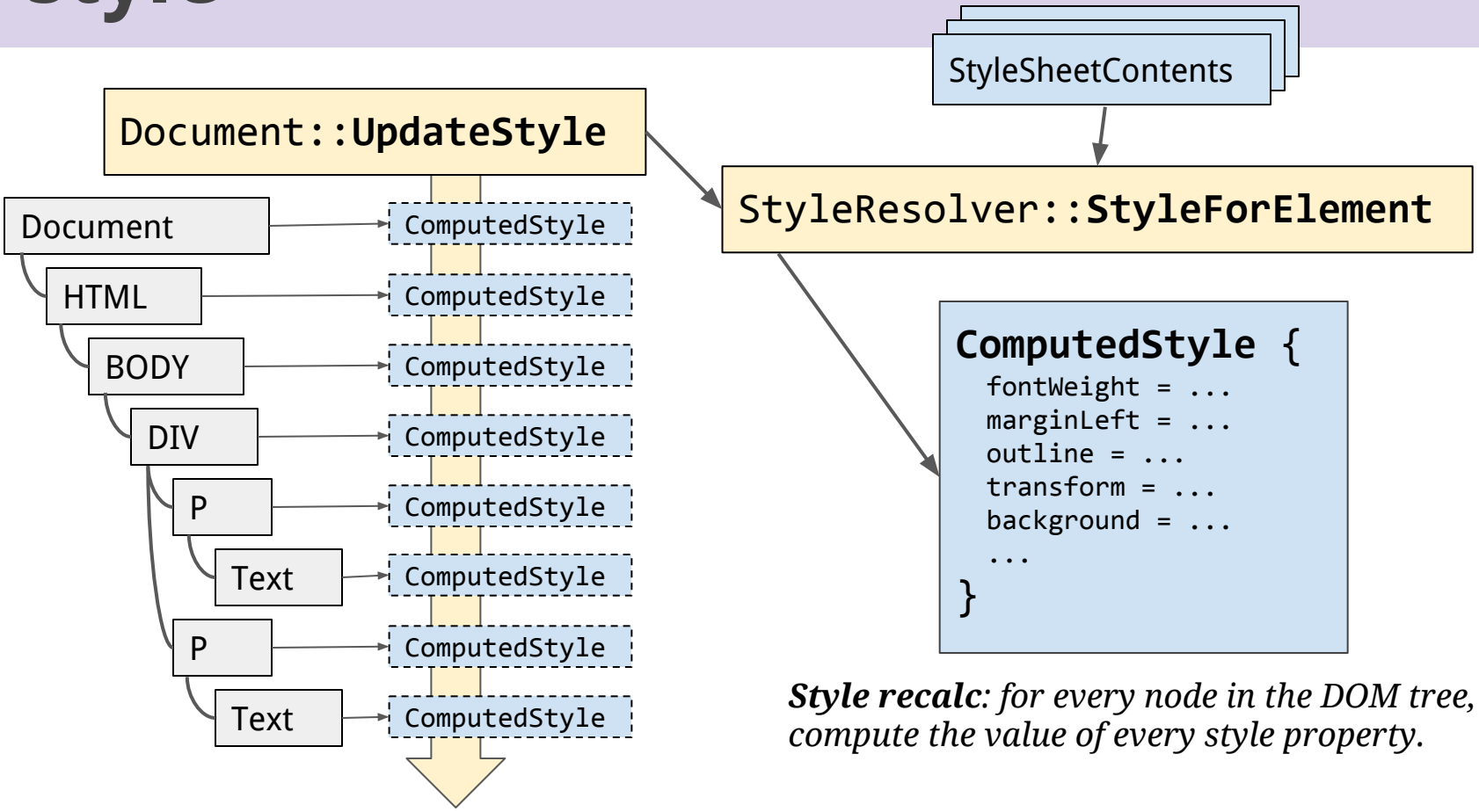
```
/* every other <p> in any <div> without class="foo" */
div:not(.foo) > p:nth-of-type(2n) {
  color: red !important;
}

p {
  color: blue;
}
```

- *selectors can be complex!*
- *declarations may conflict!*

style

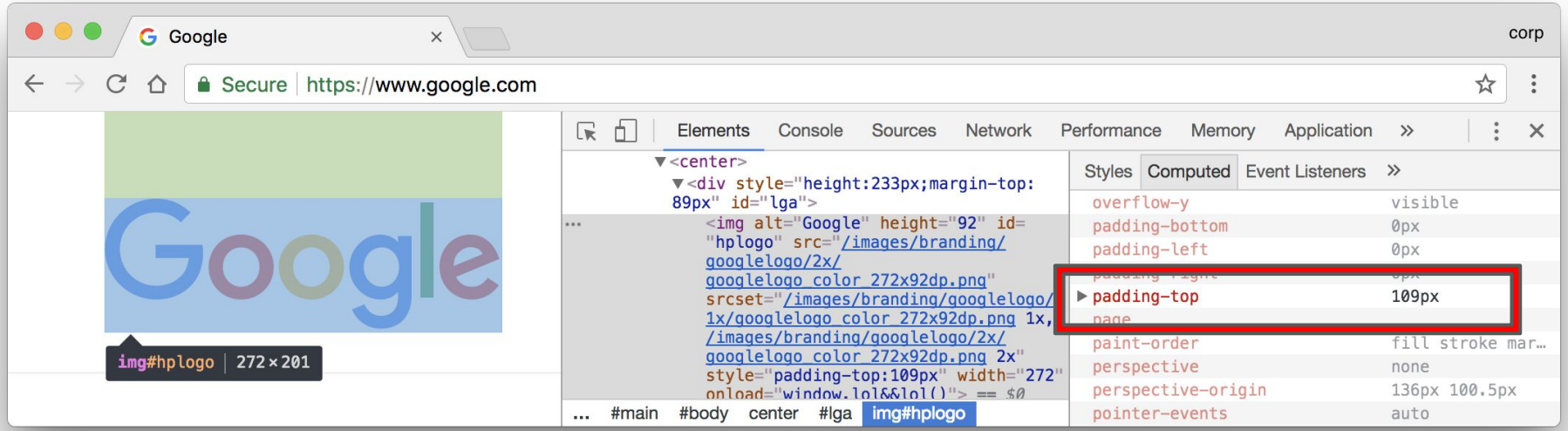

CSSParser
builds
StyleSheetContents
StyleRule
CSSSelectorList
CSSSelector
div:not(.foo) > p {
color: red !important;
}
p {
margin: 0;
border-left-color: blue;
...
}
...
CSSPropertyValueSet
CSSPropertyValue
property_
value_
CSSProperties.json5
make_css_property_subclasses.py
border_left_color.cc
BorderLeftColor
CSSColorValue
0x0000FFFF

style



***Style recalc**: for every node in the DOM tree, compute the value of every style property.*

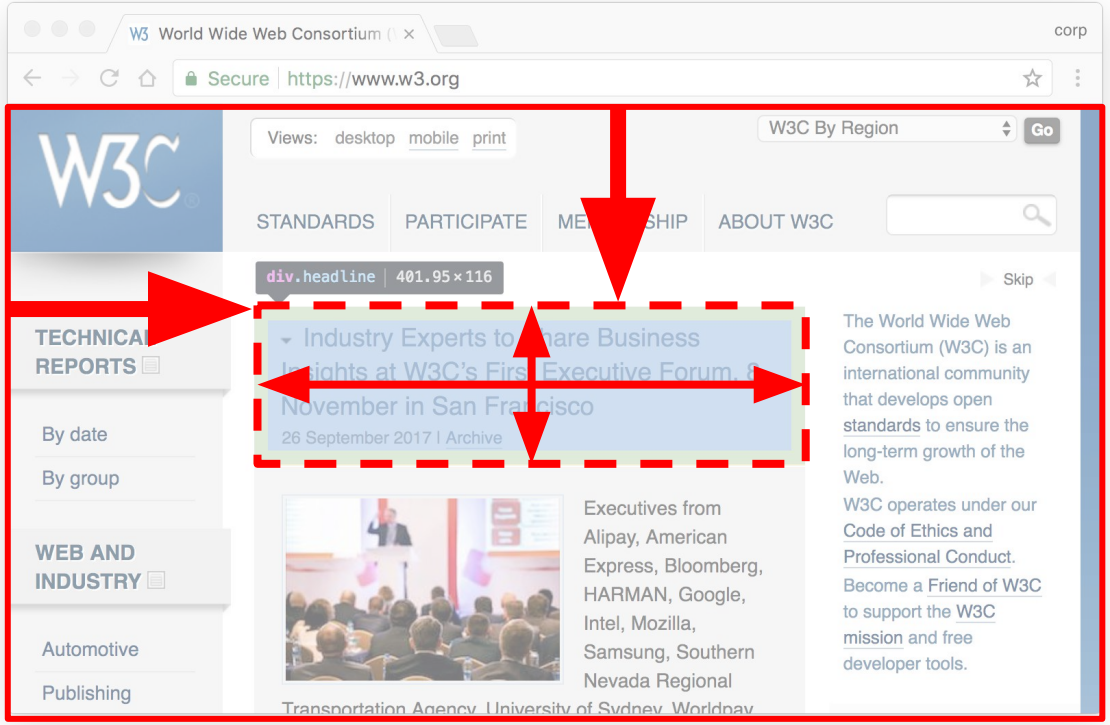
style

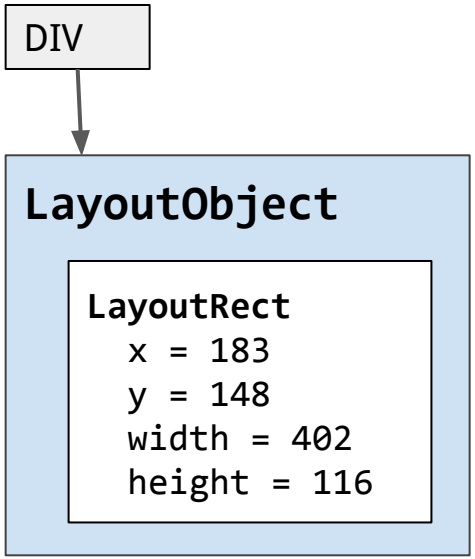



```
getComputedStyle(element)["padding-top"]
```

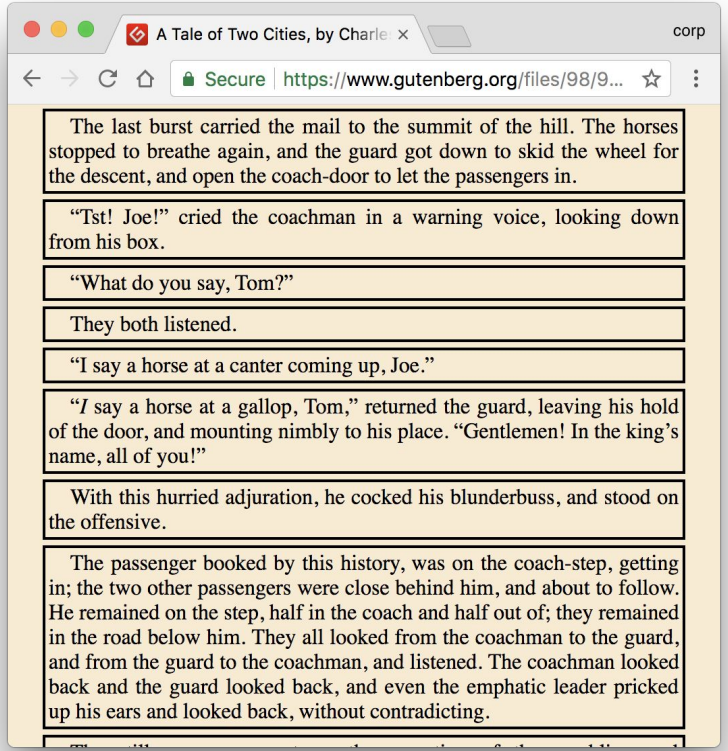
[JS]

layout


World Wide Web Consortium
corp
Secure
https://www.w3.org
W3C
Views: desktop mobile print
W3C By Region
Go
STANDARDS
PARTICIPATE
ABOUT W3C
div.headline | 401.95×116
Skip
TECHNICAL REPORTS
By date
By group
WEB AND INDUSTRY
Automotive
Publishing
Industry Experts to
November in San Fra
26 September 2017 | Archive
The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web.
W3C operates under our Code of Ethics and Professional Conduct.
Become a Friend of W3C to support the W3C mission and free developer tools.
Executives from Alipay, American Express, Bloomberg, HARMAN, Google, Intel, Mozilla, Samsung, Southern Nevada Regional

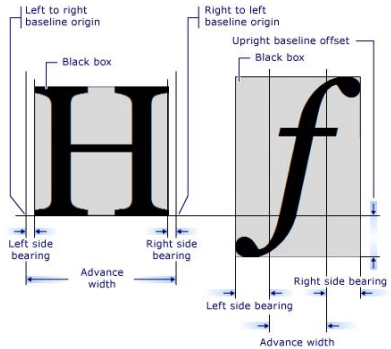

DIV
LayoutObject
LayoutRect
x = 183
y = 148
width = 402
height = 116

layout



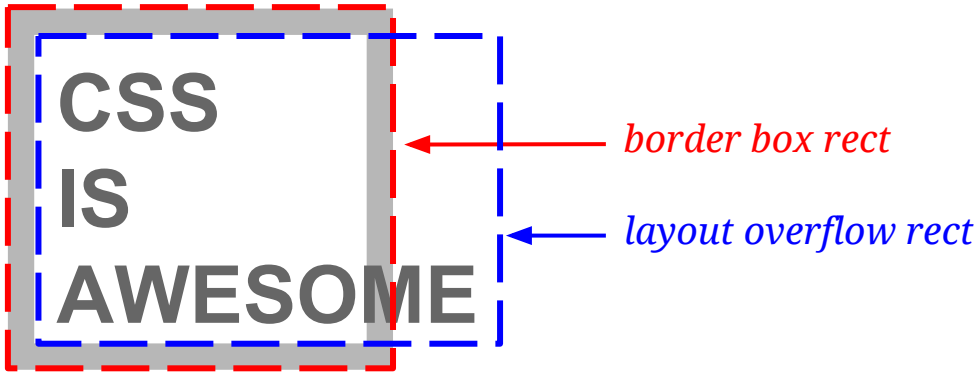
block flow

*Simple "block" layout objects are placed one after another, **flowing** down the page.*



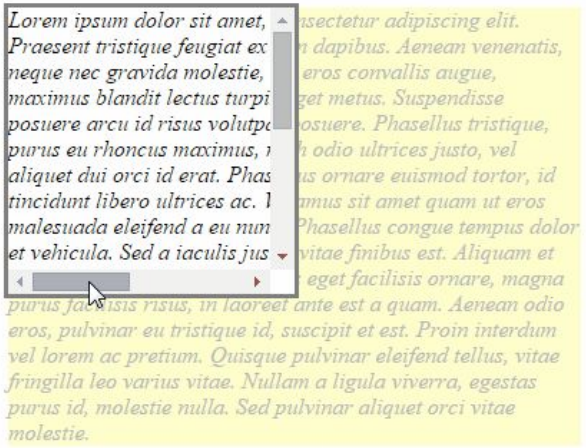
(Line breaking is not simple.)

layout



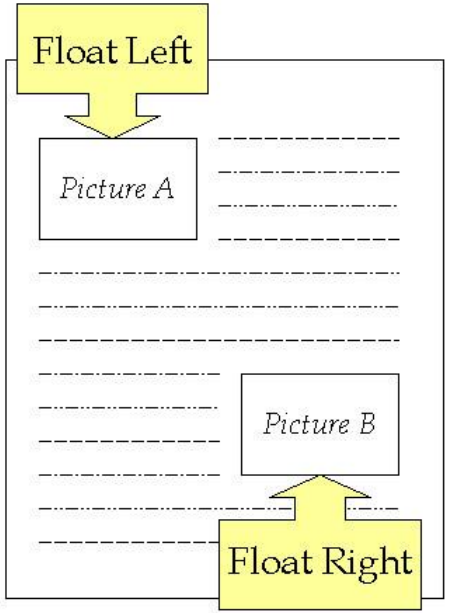
*The contents of a layout object can **overflow** its border box.*

*Overflow can be visible, hidden, or **scrollable**.*



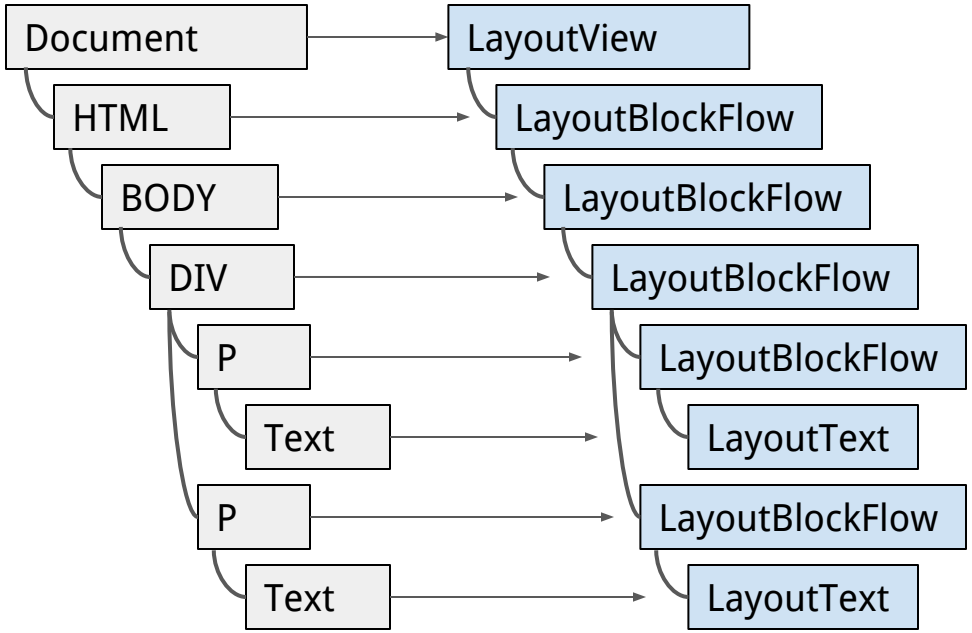
layout

Other kinds of layout are even more complex.

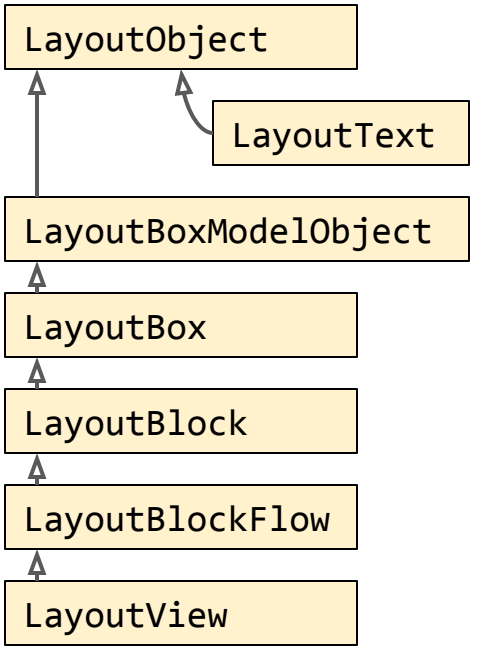


- `<table>`
- `float: left`
- `column-count: 3`
- `display: flex`
- `writing-mode: vertical-lr`
- `...`

layout



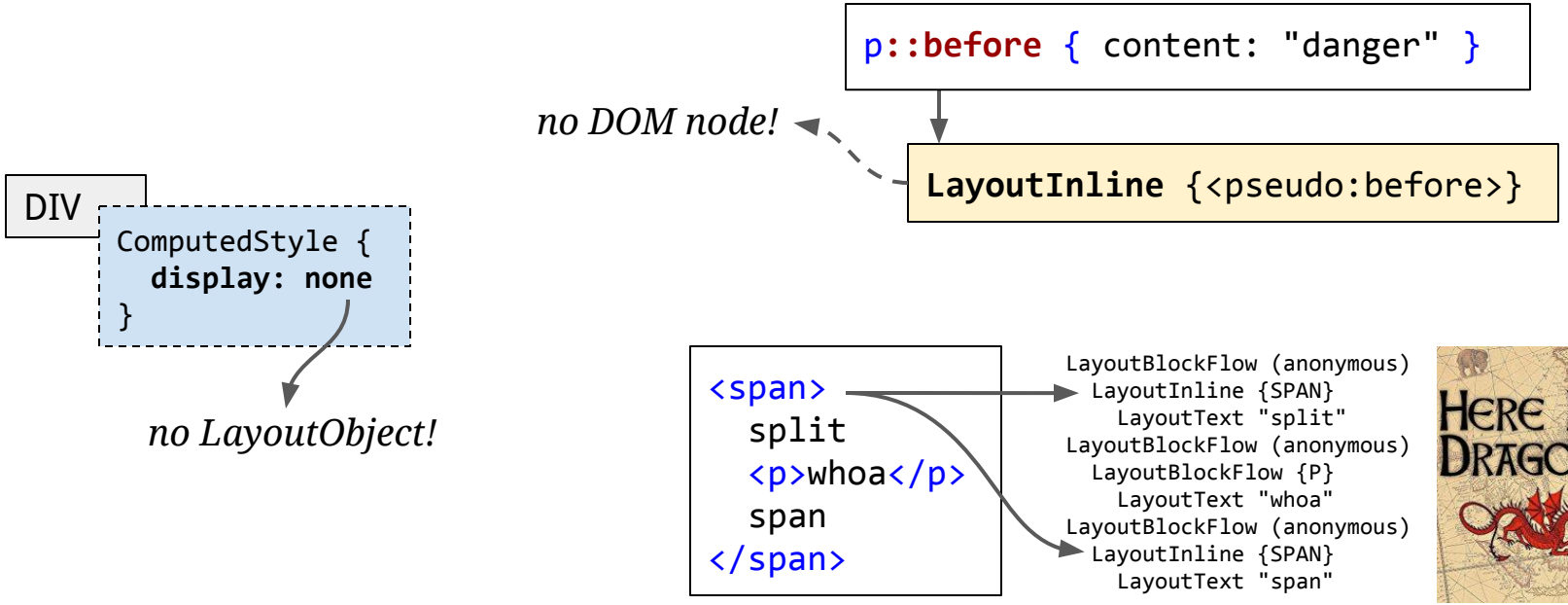
DOM tree *layout tree*



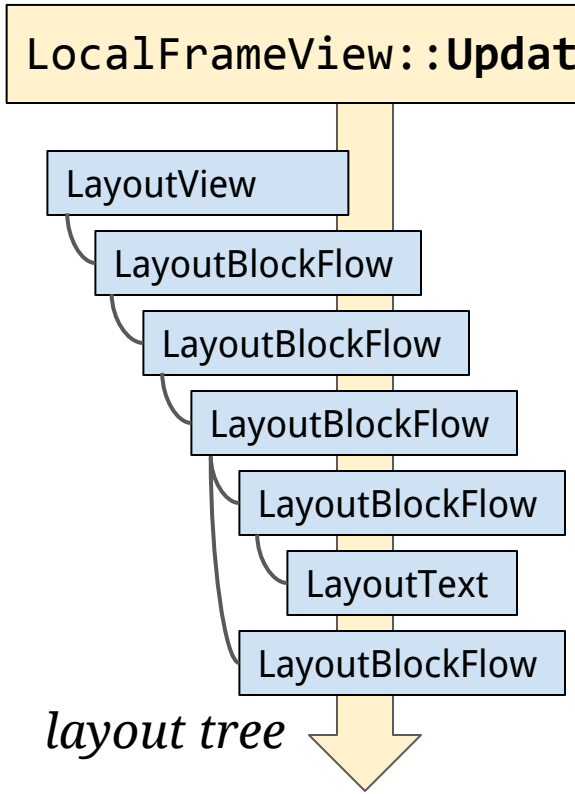
C++ class hierarchy

layout

*DOM nodes are **mostly** 1:1 with layout objects, with some exceptions.*



layout

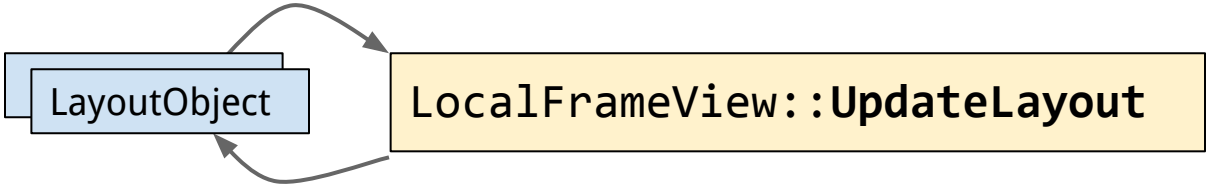


`LayoutObject::UpdateLayout`

- *compute box bounds and overflow*
- *create line boxes*
- *add/remove scrollbars*

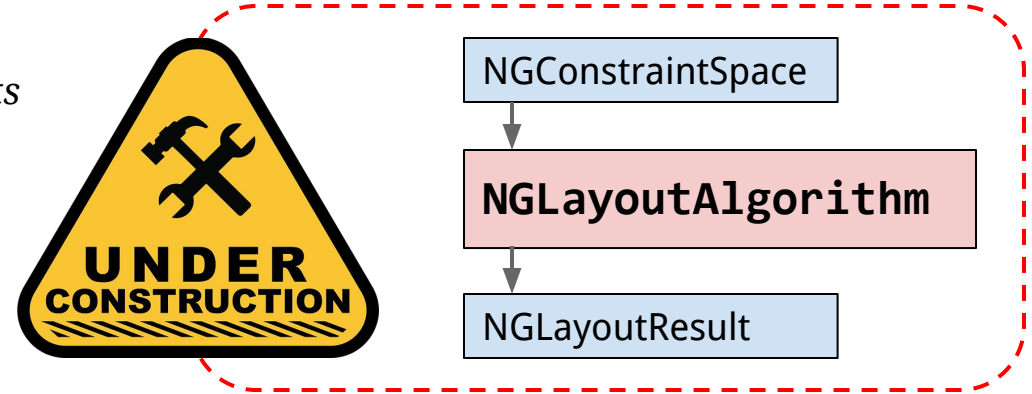
Layout *(lifecycle stage) walks the* ***layout tree*** *filling in state describing the visual geometry of each* `LayoutObject`.

layout

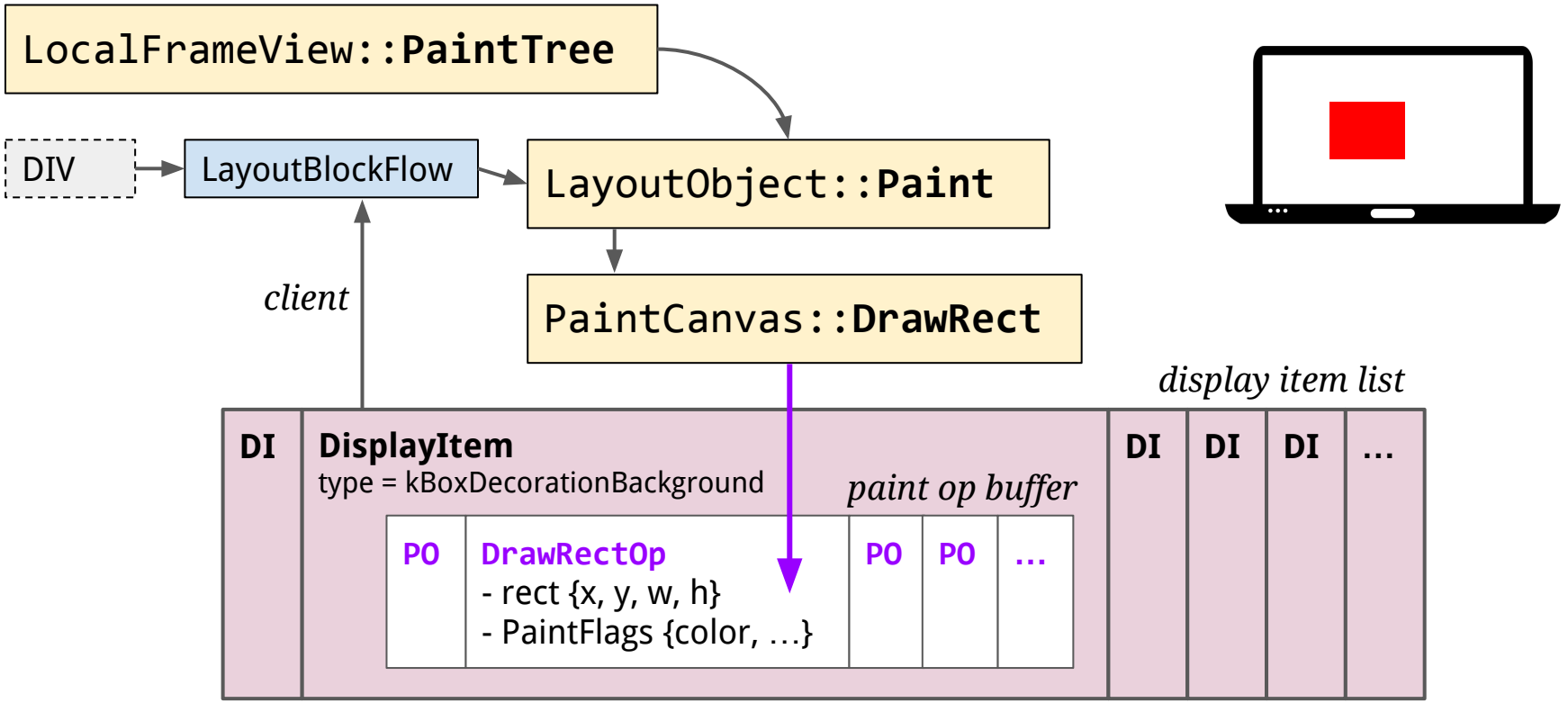


(today)

*A new system called **LayoutNG** will separate inputs from outputs more cleanly.*



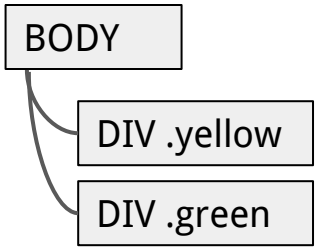
paint


LocalFrameView::PaintTree
DIV
LayoutBlockFlow
LayoutObject::Paint
client
PaintCanvas::DrawRect
display item list
DI
DisplayItem
type = kBoxDecorationBackground
paint op buffer
PO
DrawRectOp
- rect {x, y, w, h}
- PaintFlags {color, …}
PO
PO
…
DI
DI
DI
…

paint

*Paint uses **stacking order**, not DOM order.*

```
<div class="yellow"></div>
<div class="green"></div>
<style>
  .yellow { z-index: 2; ... }
  .green  { z-index: 1; ... }
</style>
```



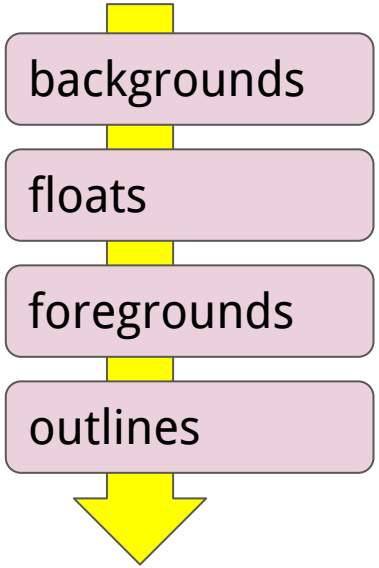
DOM tree

yellow paints last

paint

*Each **paint phase** is a separate traversal of a stacking context.*

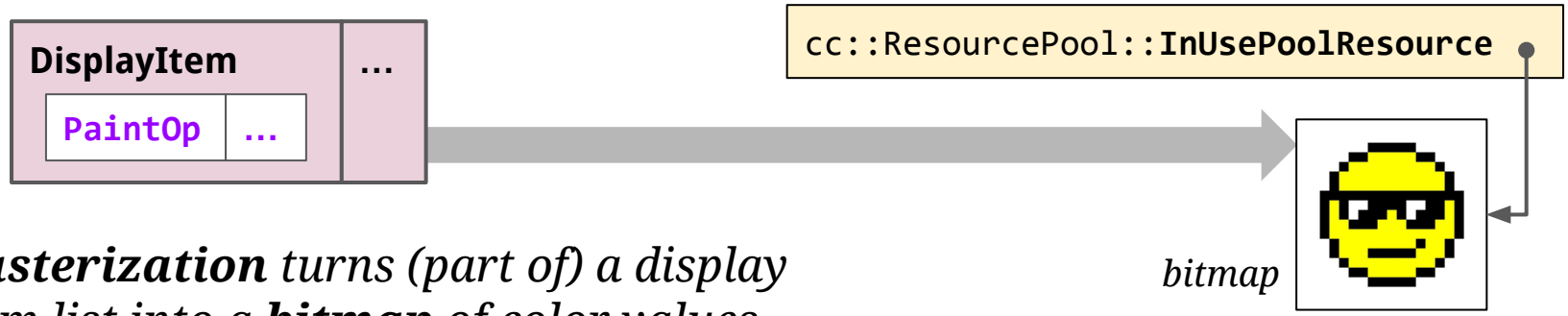
paint phases (simplified)




```
<div id="green">
  green's text
</div>
<div id="blue"></div>
```

blue after green, but foregrounds after backgrounds

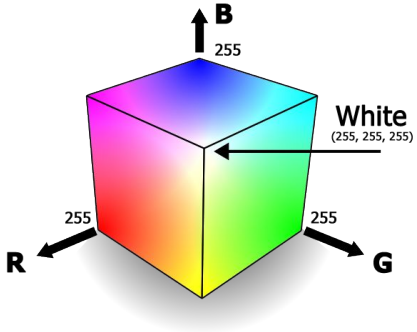
raster



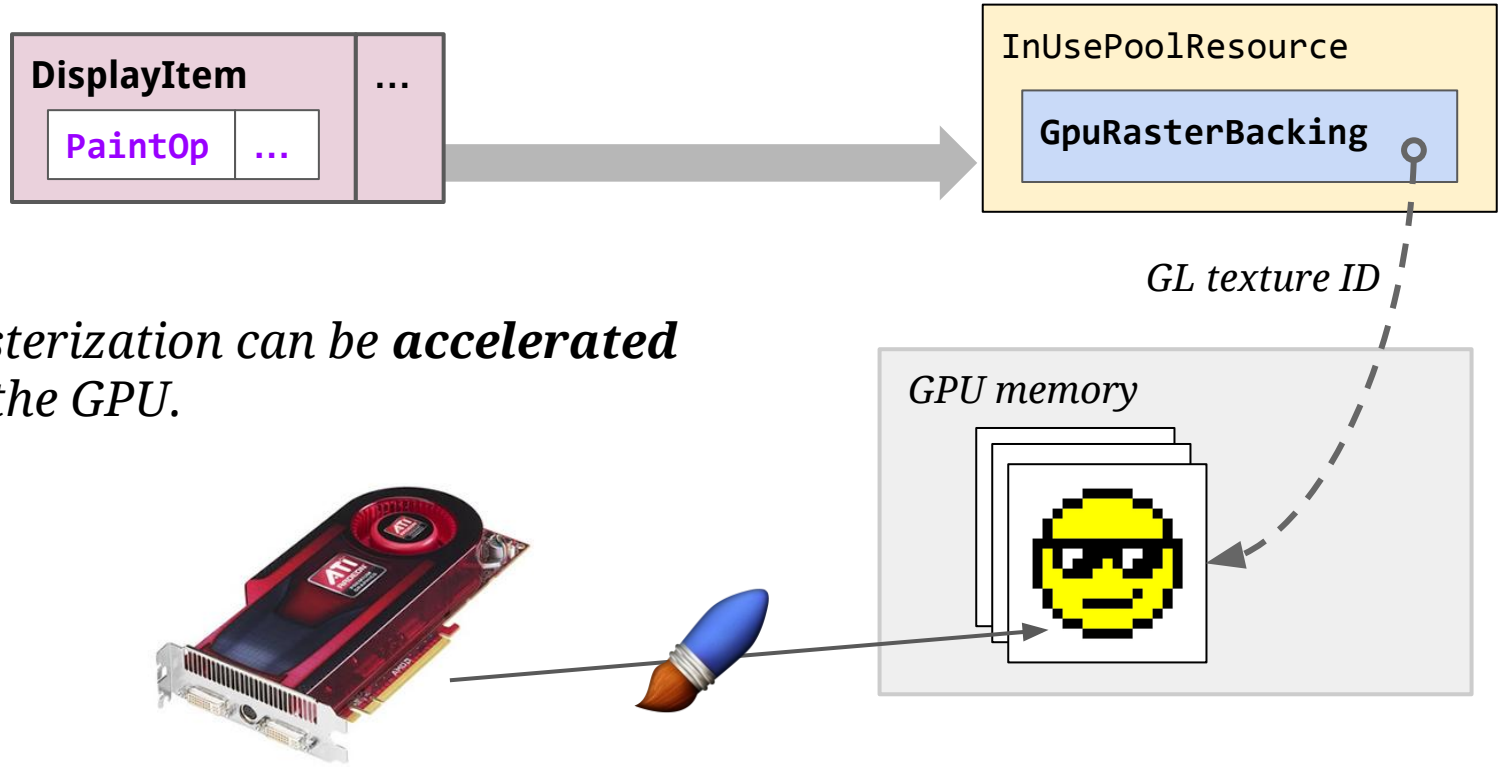
Rasterization *turns (part of) a display item list into a* ***bitmap*** *of color values.*

FFFF FFFF	FFFF FFFF	0000 00FF
FFFF FFFF	0000 00FF	FFFF 00FF
0000 00FF	FFFF 00FF	FFFF 00FF

(red, green, blue, alpha)

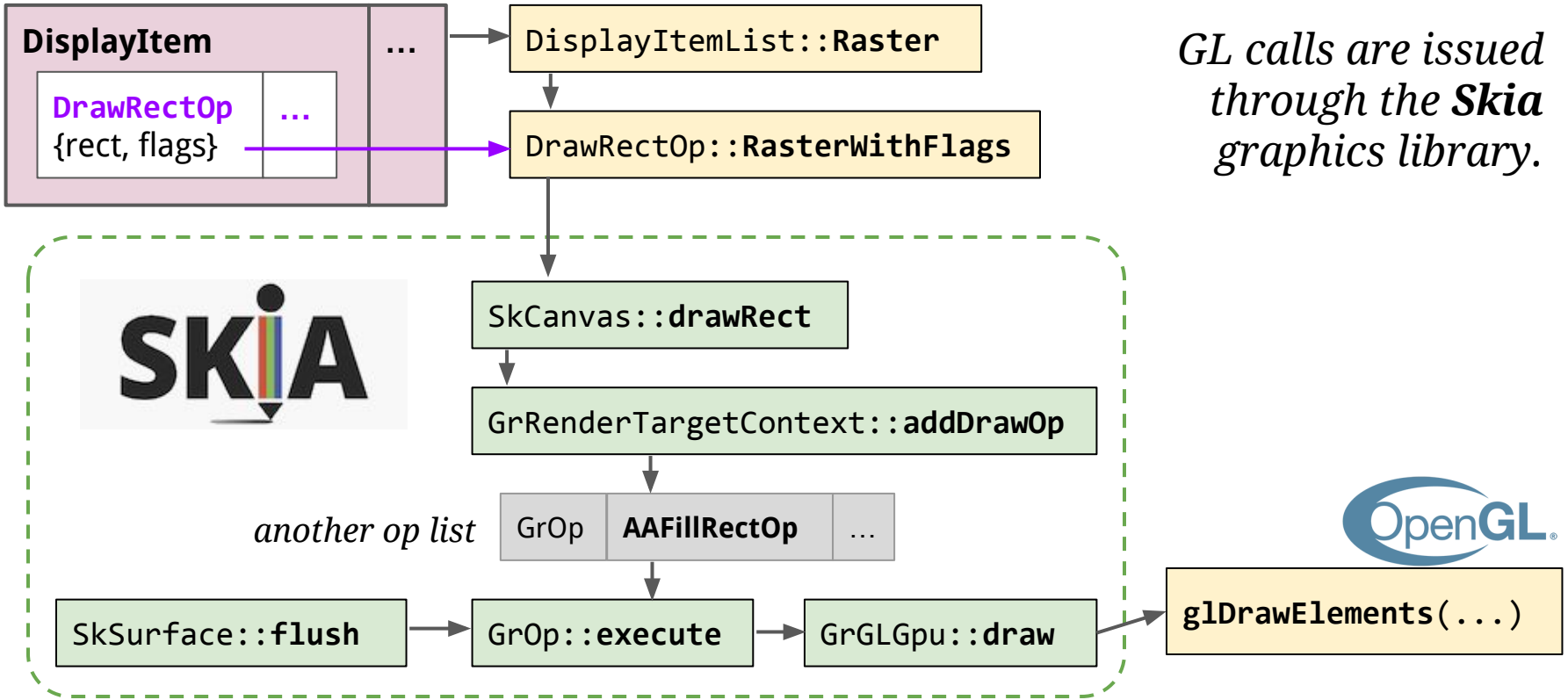


raster



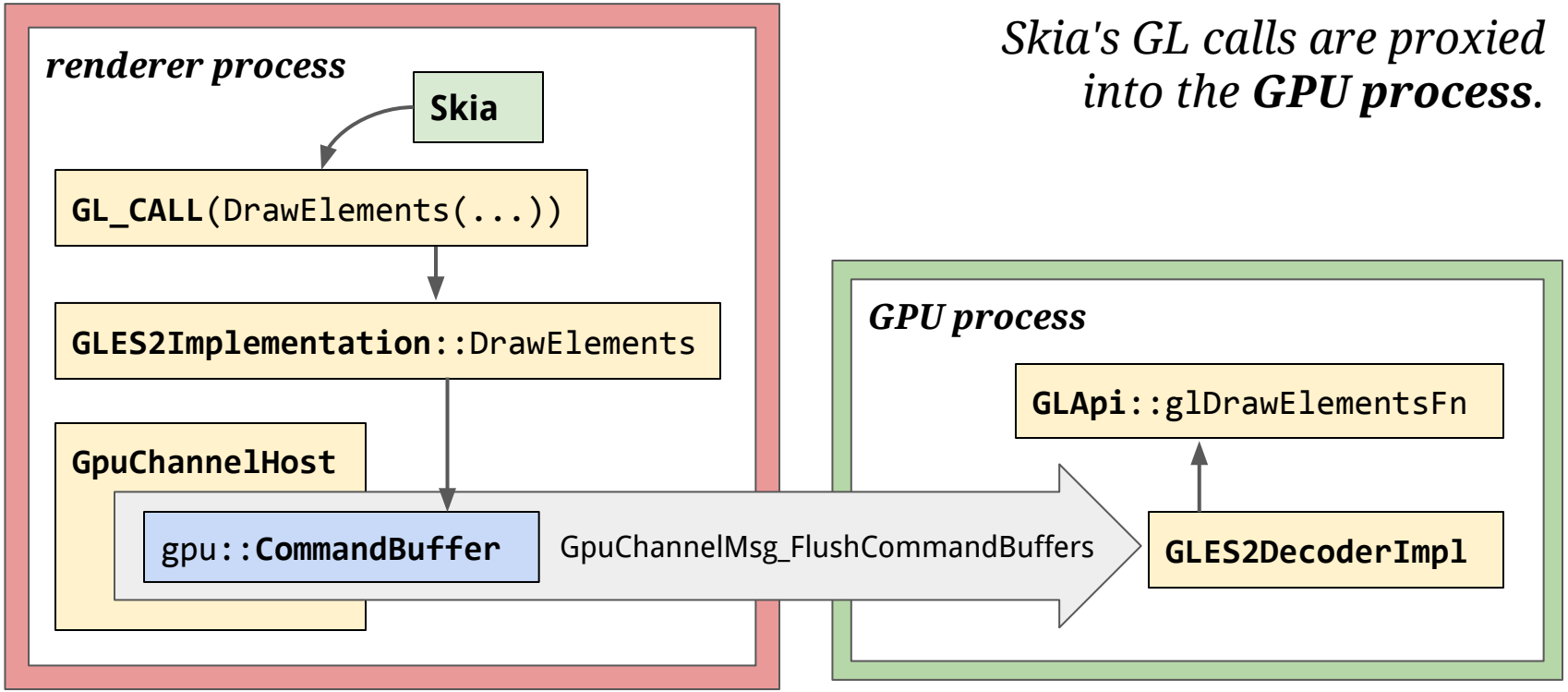
*Rasterization can be **accelerated** by the GPU.*

raster



*GL calls are issued through the **Skia** graphics library.*

gpu



*Skia's GL calls are proxied into the **GPU process**.*

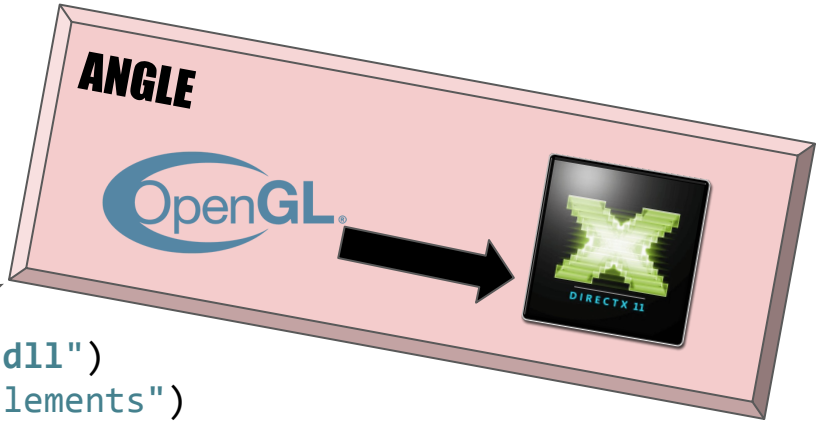
gpu

GPU process

The GL functions are dynamically linked.
On Windows, we translate to DirectX.

```
GLApi::glDrawElementsFn
```

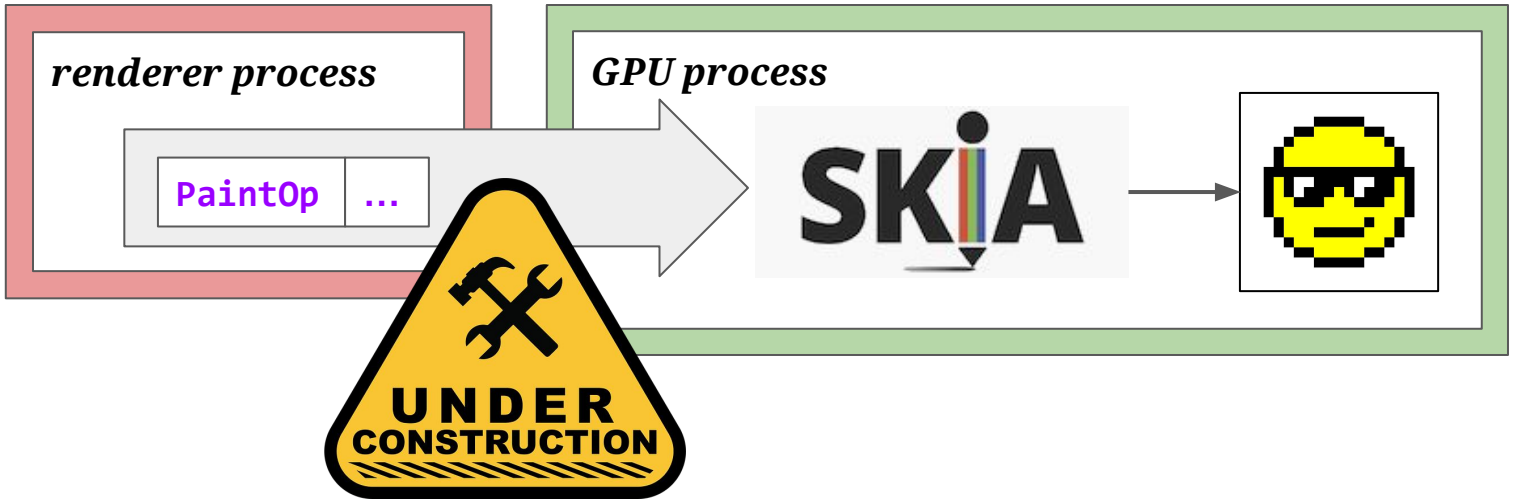
```
dlopen("libGLESv2.so")
dlsym("glDrawElements")
```



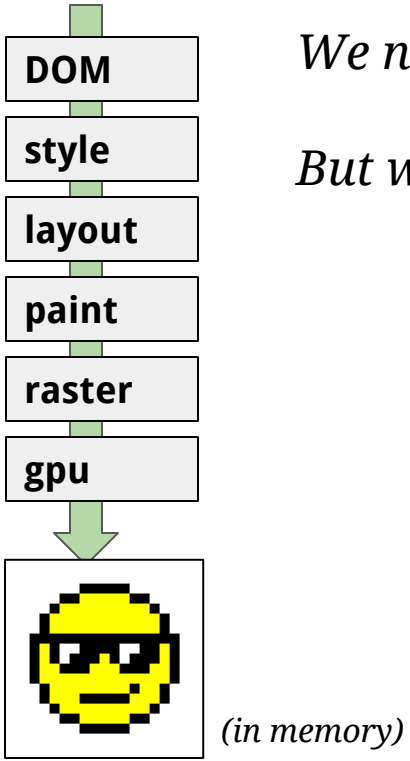

```
LoadLibrary("libglesv2.dll")
GetProcAddress("glDrawElements")
```

gpu

In the future, raster will happen in the GPU process.



change



(in memory)

*We now have a complete **pipeline**.*

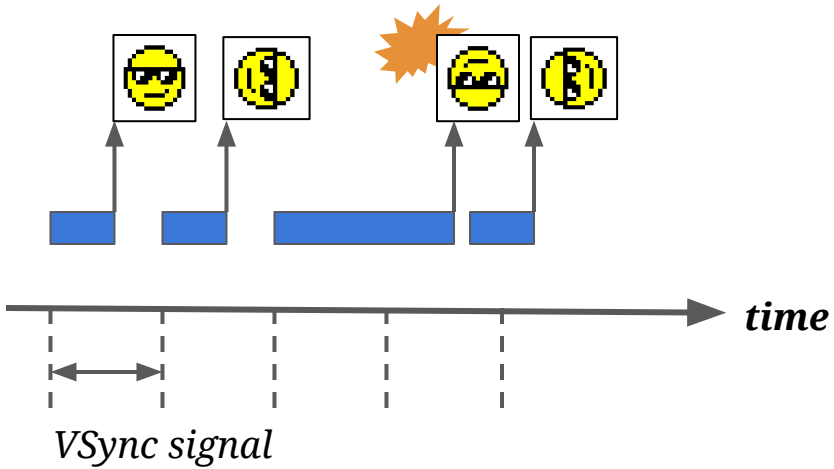
*But what if state can **change**?*

- scrolling
- zooming
- animations
- incremental loading
- JavaScript
- …

"Change is good."
"Yeah, but it's not easy."

frames

*The renderer produces **animation frames**.*

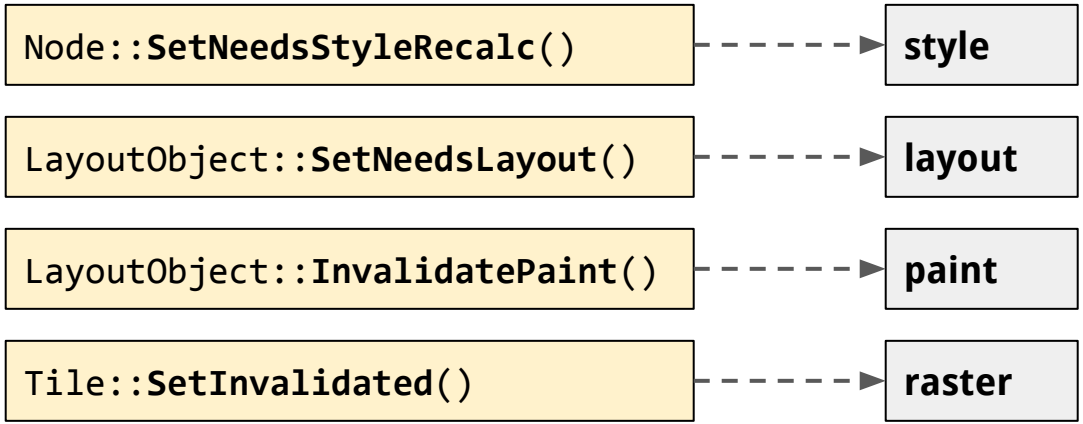


*Below 60 frames per second, scrolling and animations look "**janky**".*

= *time spent to render the frame*

invalidation

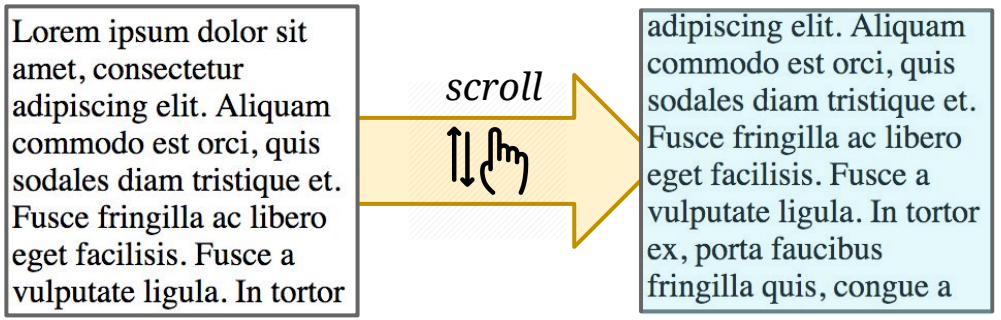
*Each pipeline stage tracks granular asynchronous **invalidations**.*



Outputs are reused from previous frames when possible.

repaint

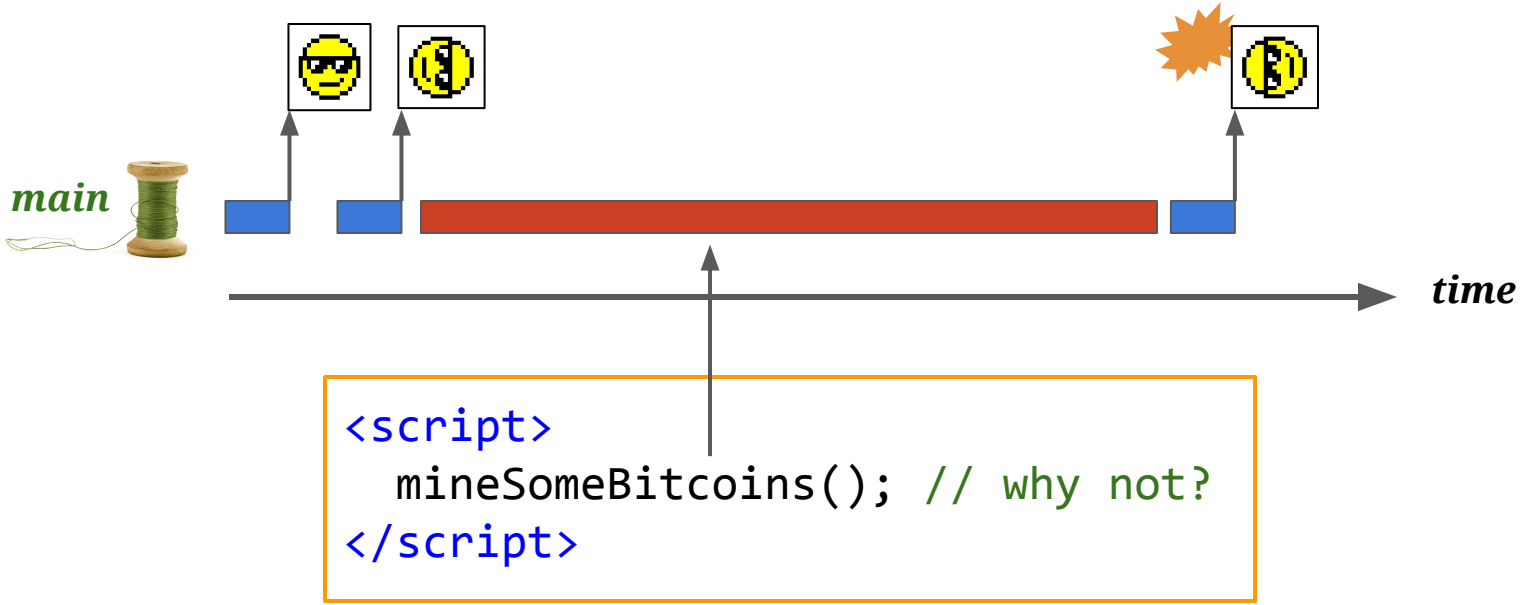
Paint + raster remain expensive if a large region is transformed…



all the pixels changed!

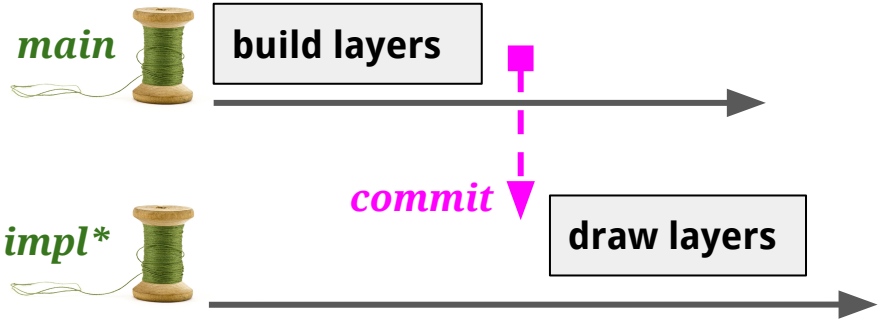
jank

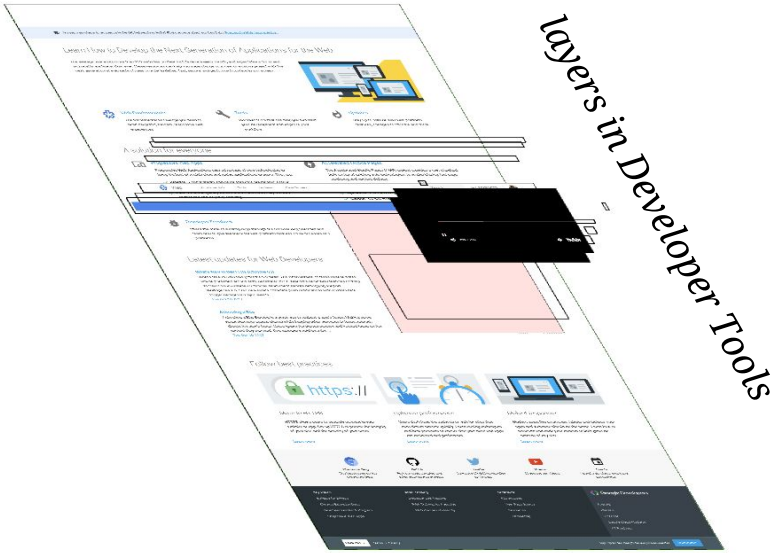
*... and anything on the **main thread** competes with JavaScript.*



enter: compositing

- *Decompose the page into **layers**.*
- ***Composite** the layers on another thread.*



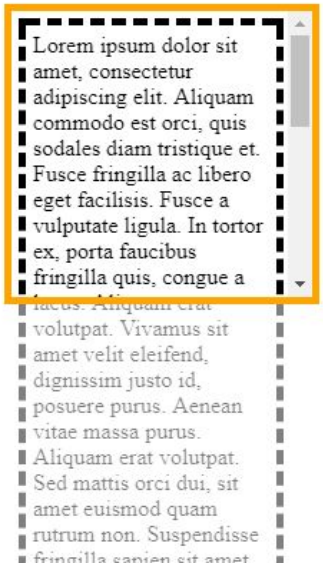


** ("impl" = compositor thread) ¯_(ツ)_/¯*

compositing

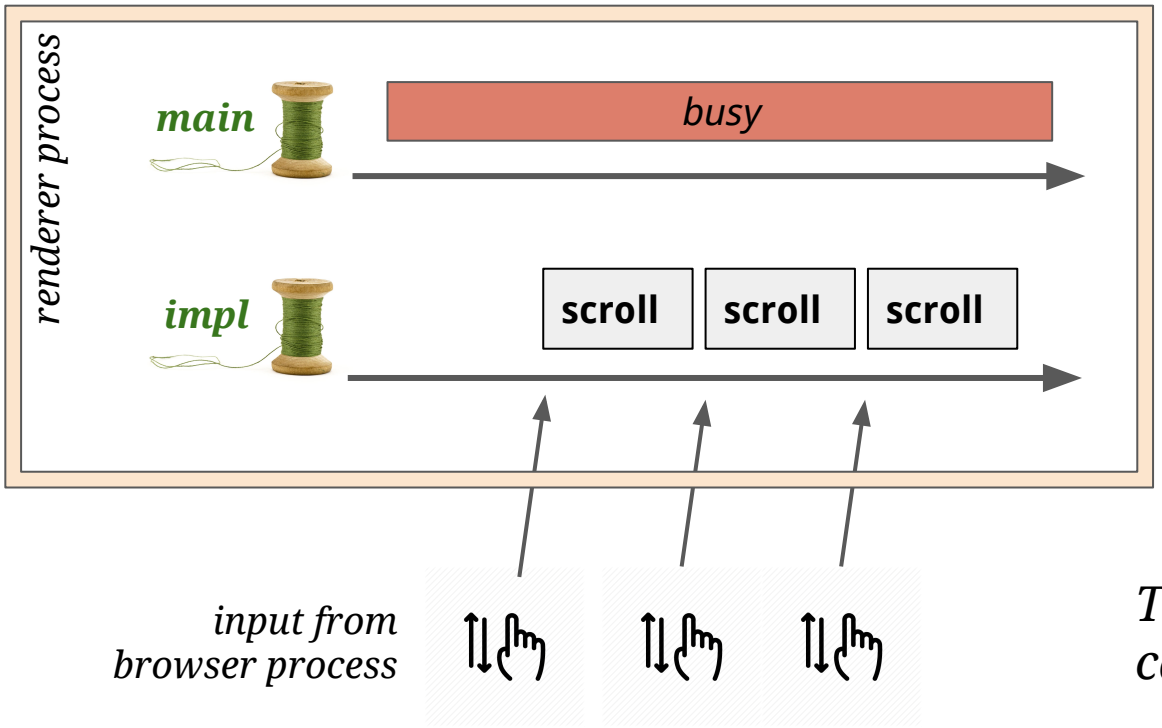
Animation:
a layer moves

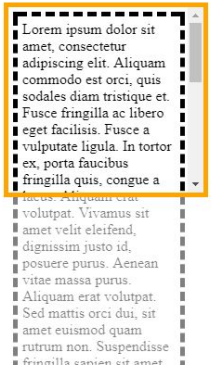
Scrolling:
a layer moves;
another clips



Pinch Zoom:
a layer scales

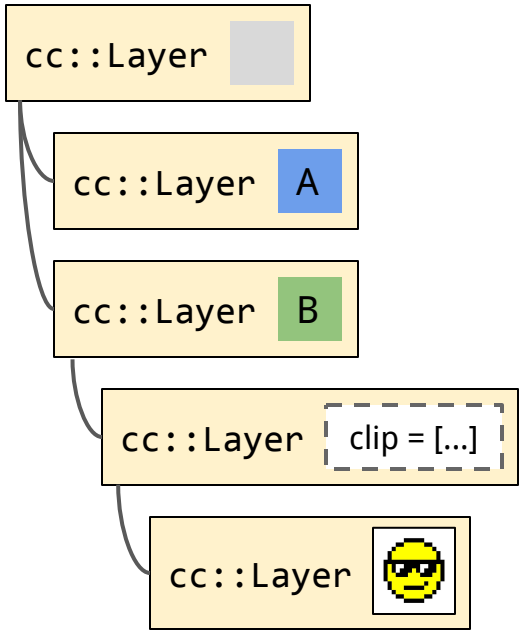
compositing





*The compositor thread can handle **input** events.*

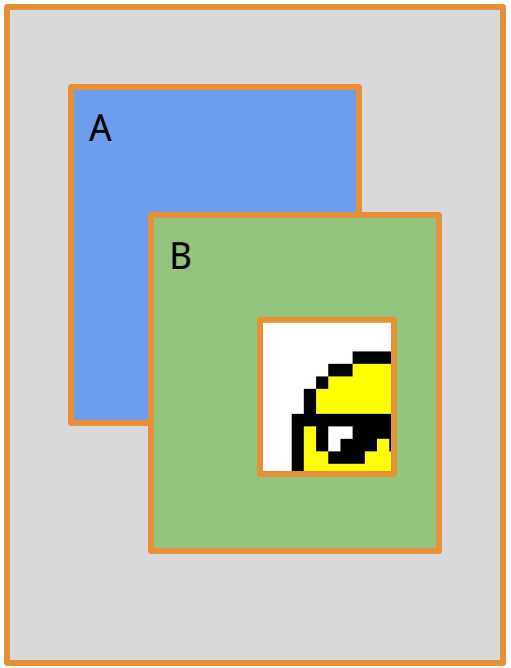
layer tree



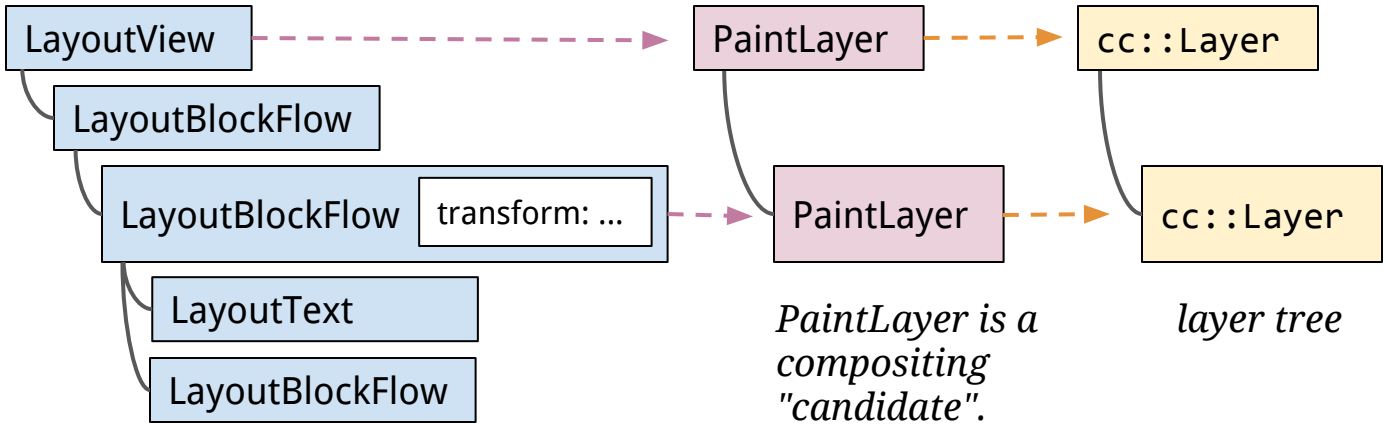
layer tree

Layers are also a tree!

- *"stacked" by preorder traversal*
- *can clip descendants, or apply effects*

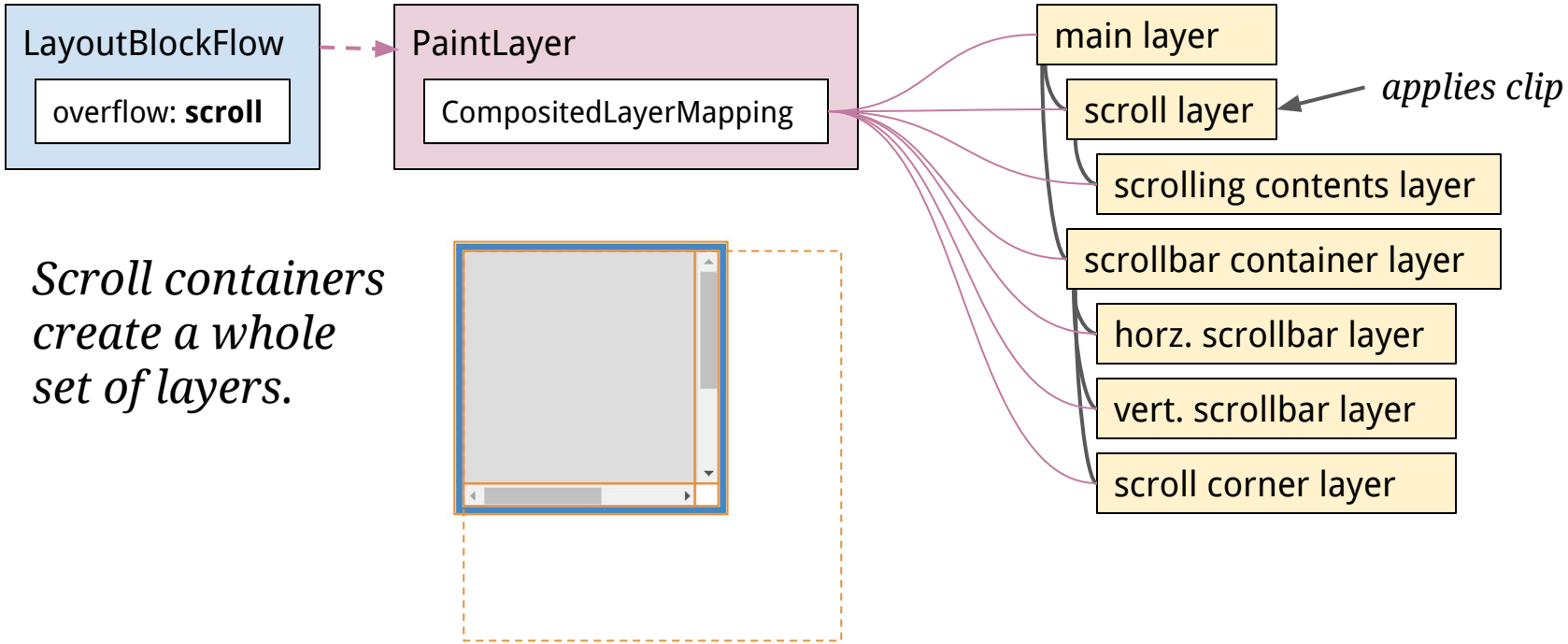


layer tree



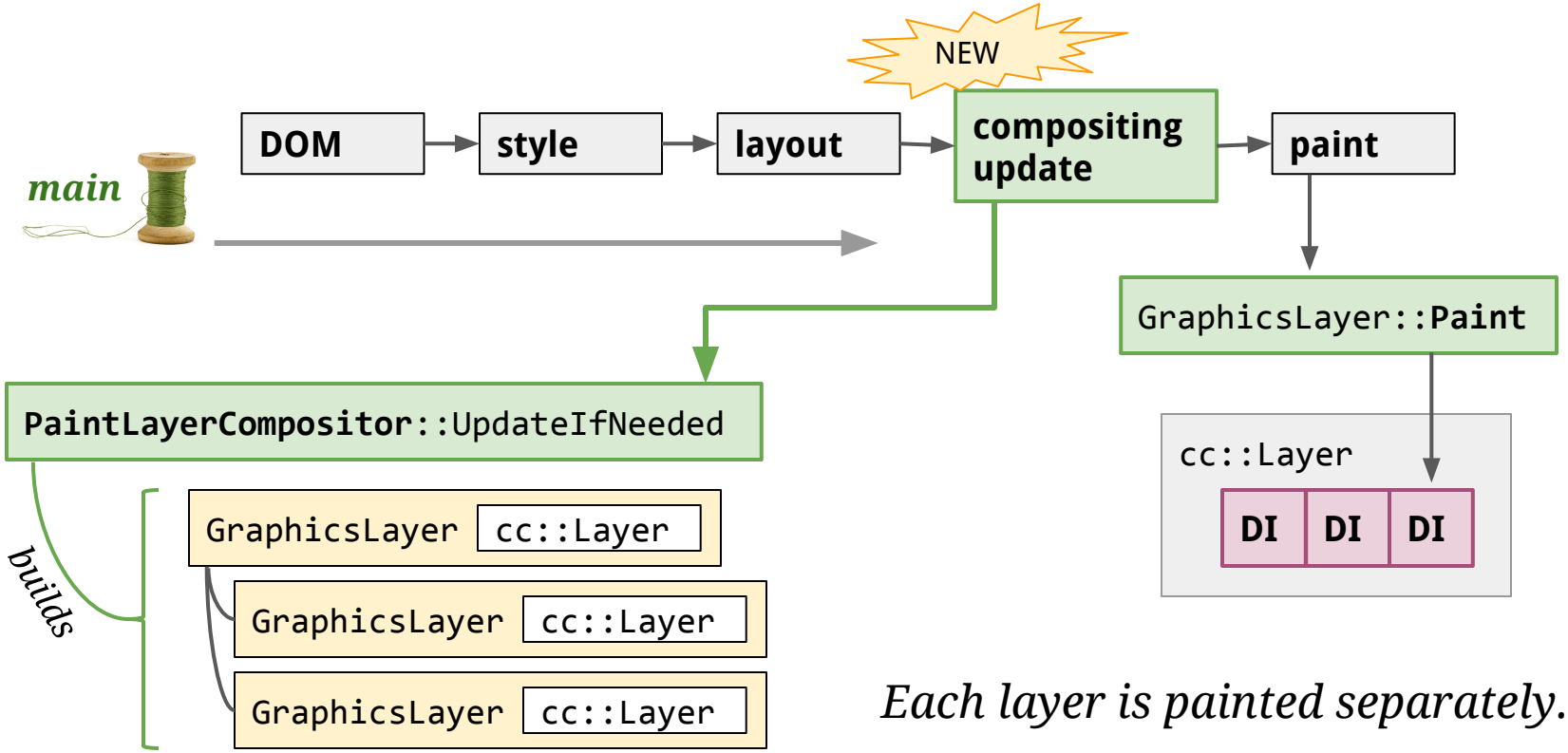
The layer tree is based on the layout tree.

layer tree



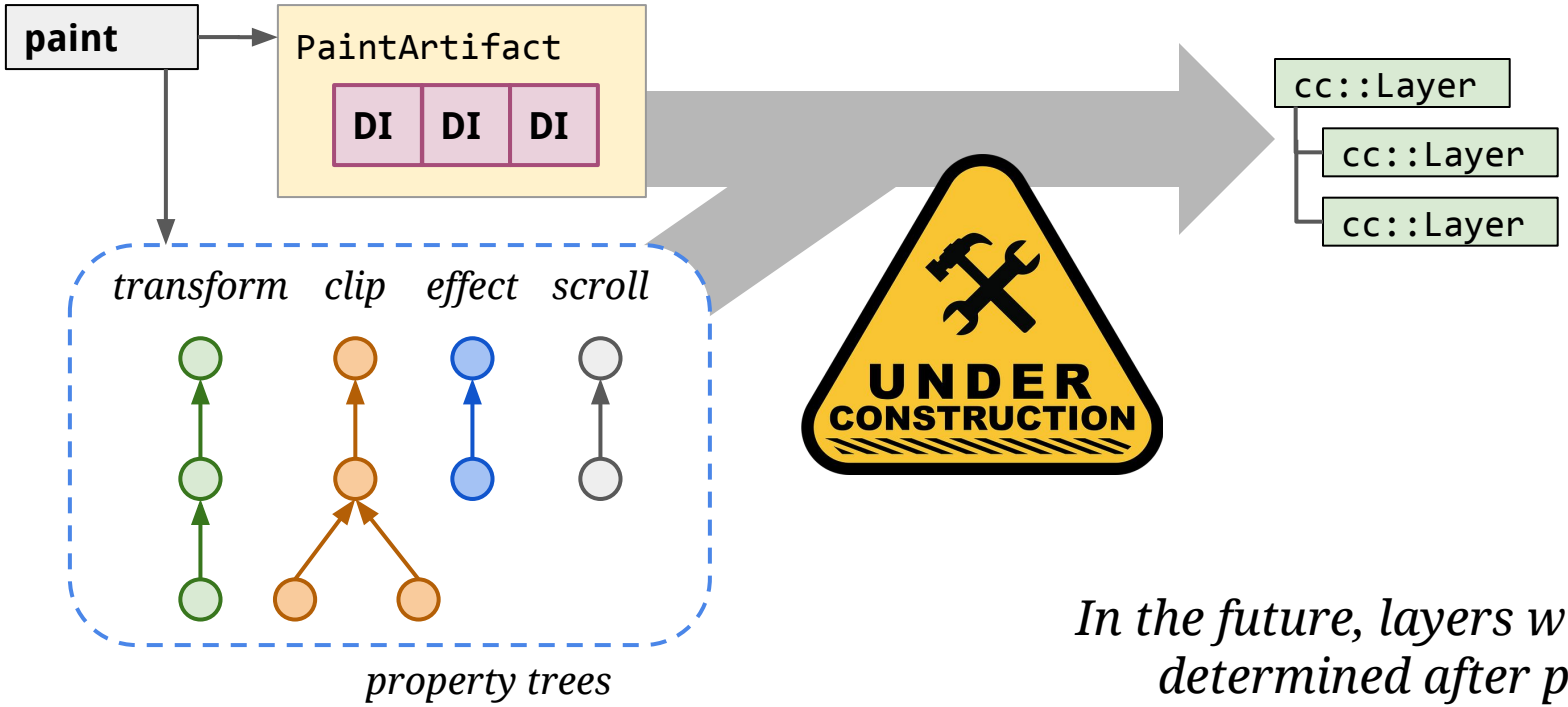
Scroll containers create a whole set of layers.

compositing update



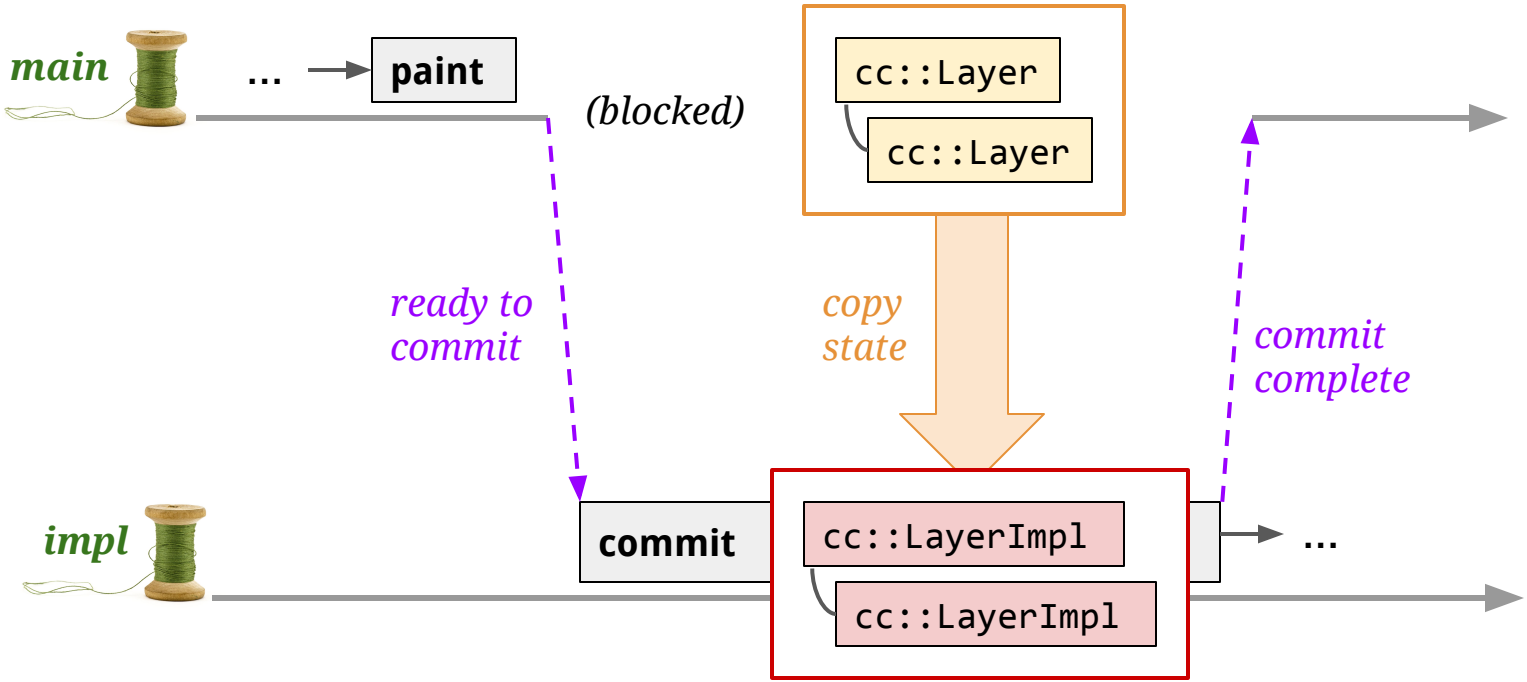
Each layer is painted separately.

slimming paint

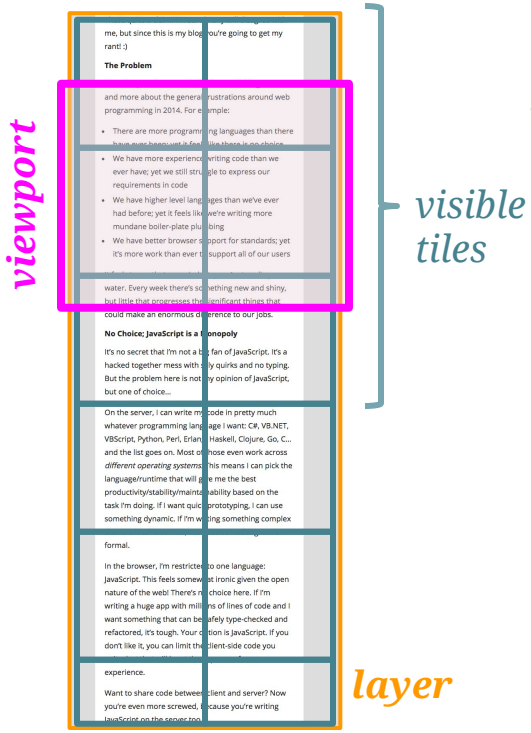


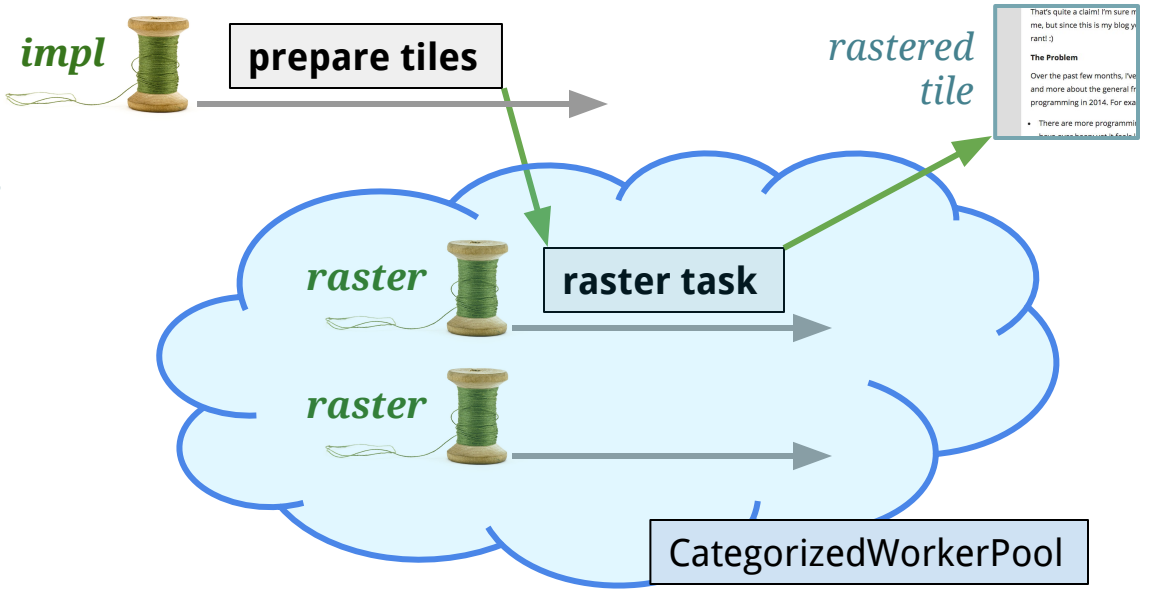
In the future, layers will be determined after paint.

commit


main
...
paint
(blocked)
cc::Layer
cc::Layer
ready to commit
copy state
commit complete
impl
commit
cc::LayerImpl
cc::LayerImpl
...

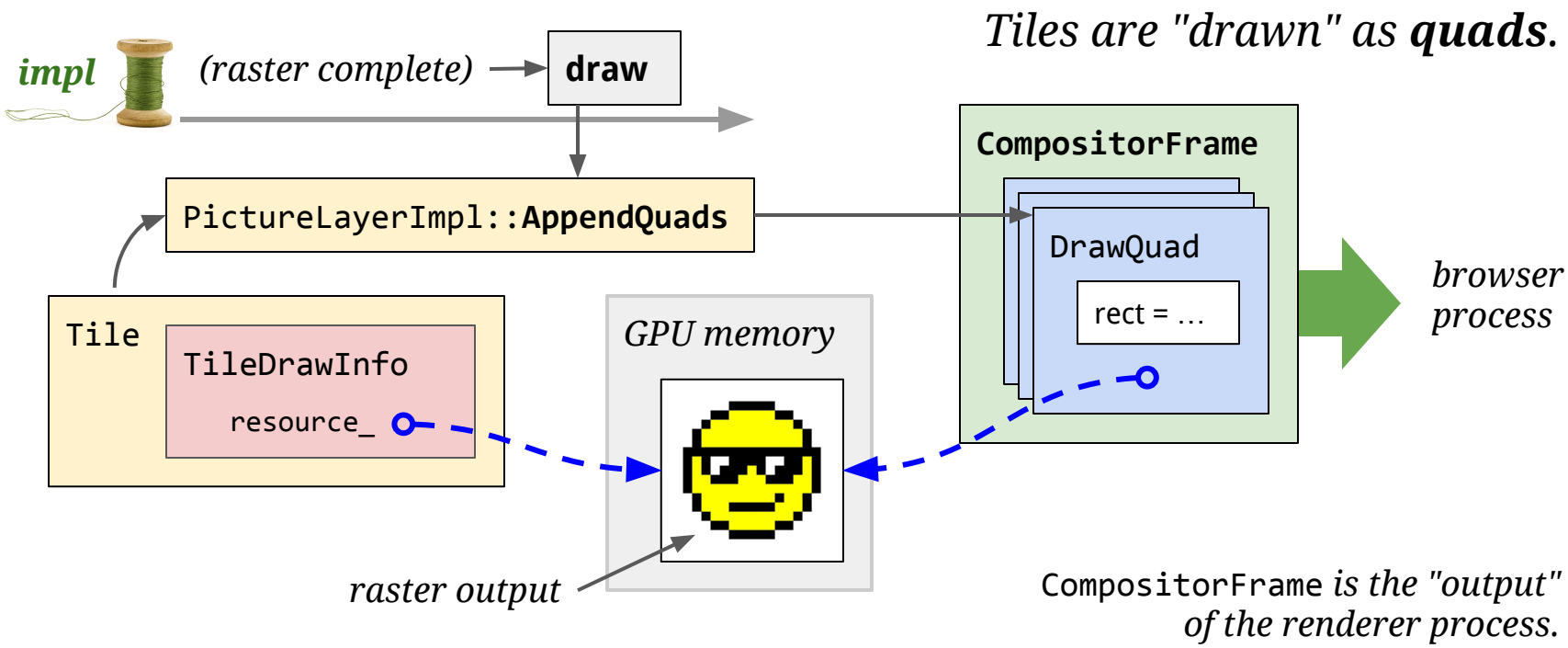
tiling



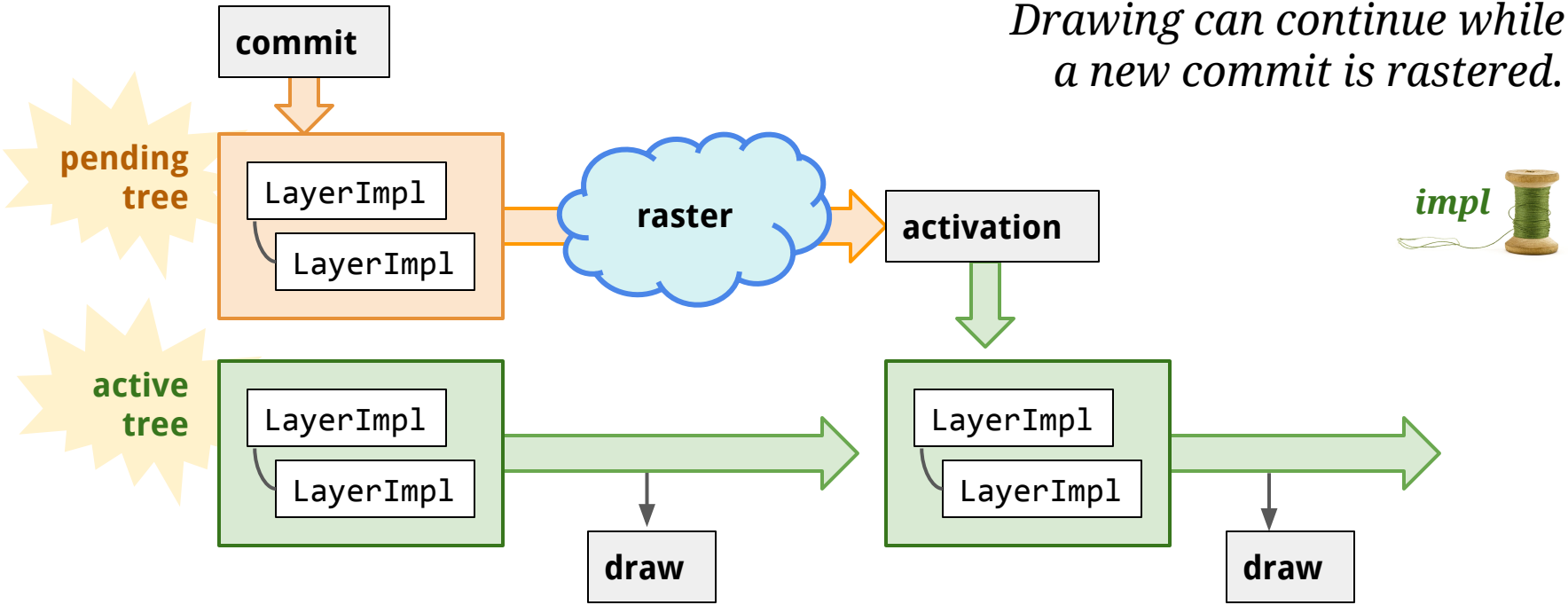


*Layers are broken into **tiles** for raster.*

drawing

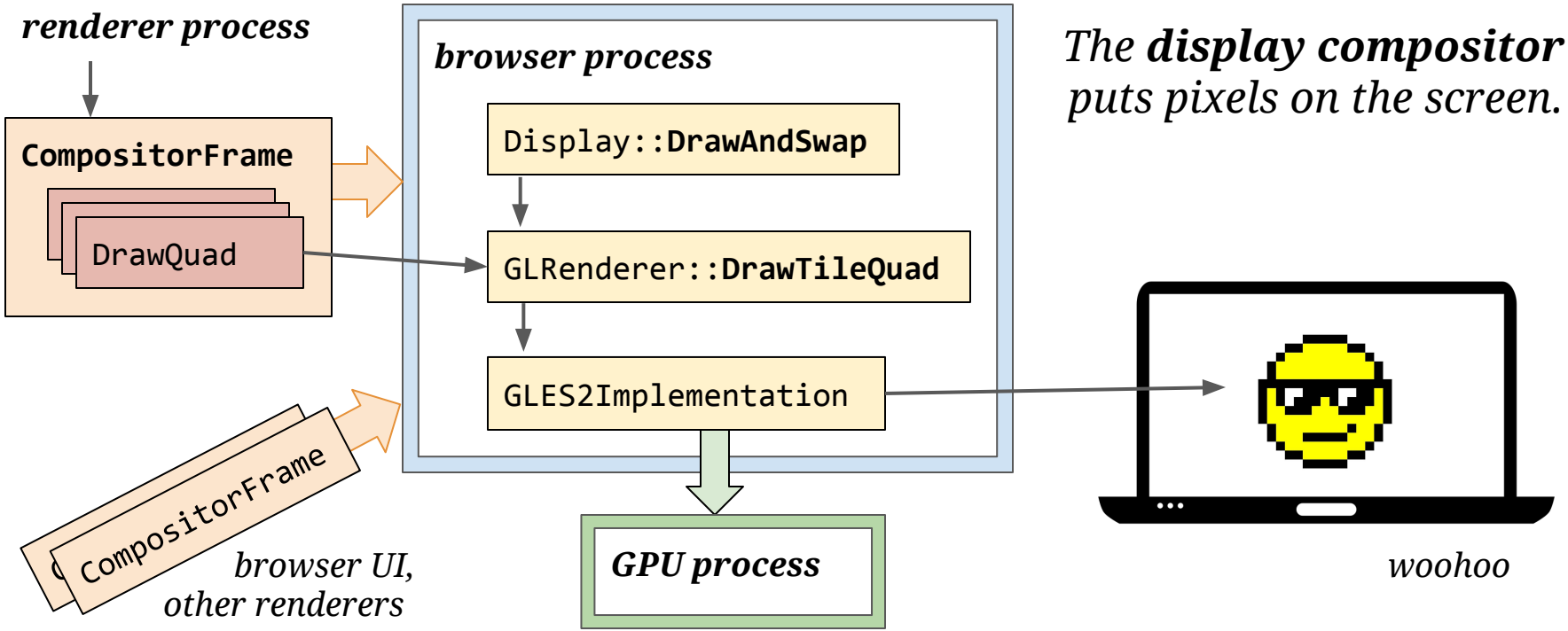

impl
(raster complete)
draw
Tiles are "drawn" as quads.
CompositorFrame
PictureLayerImpl::AppendQuads
DrawQuad
rect = …
browser process
Tile
TileDrawInfo
resource_
GPU memory
raster output
CompositorFrame is the "output" of the renderer process.

activation

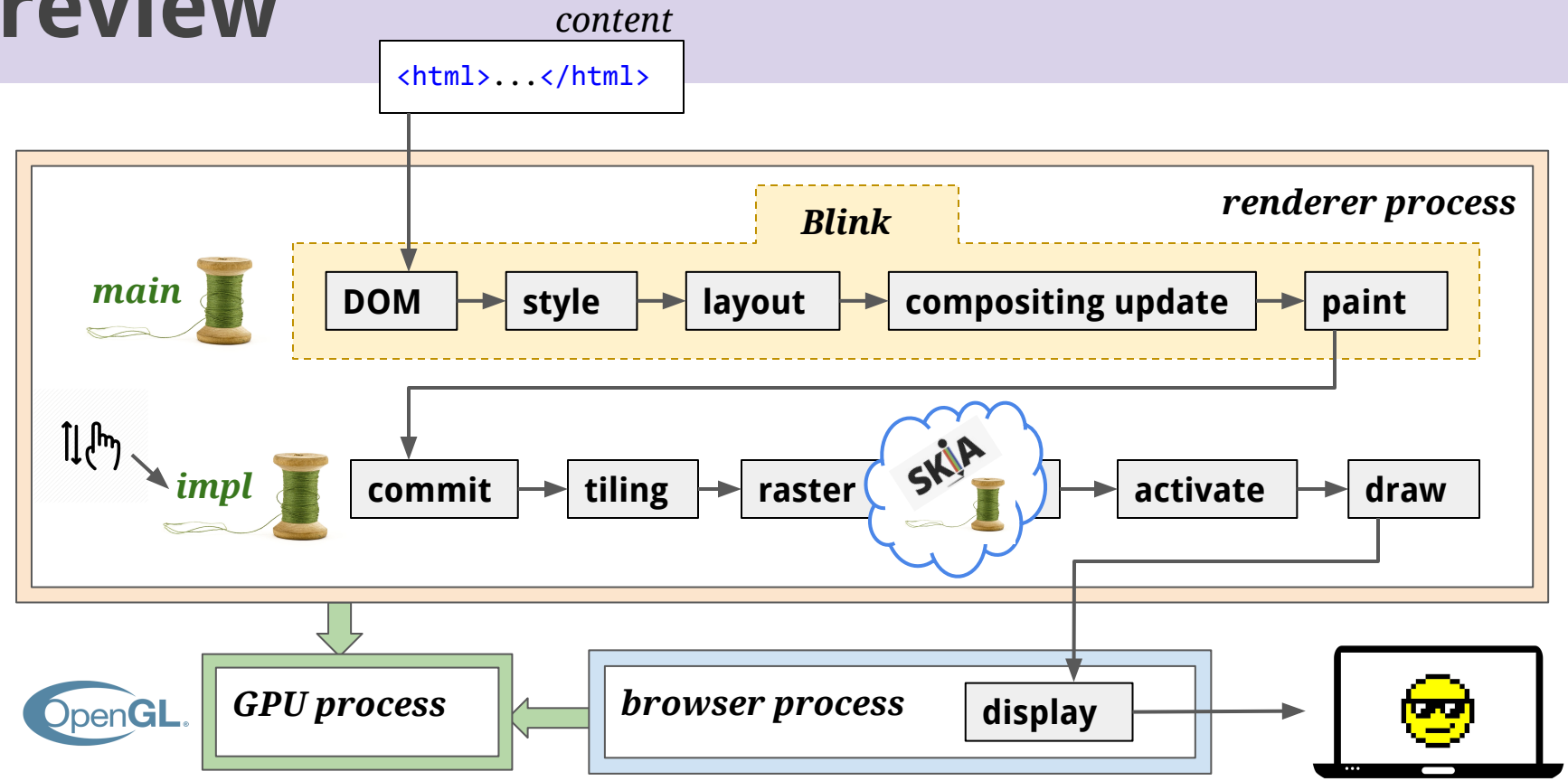


Drawing can continue while a new commit is rastered.

display (viz)

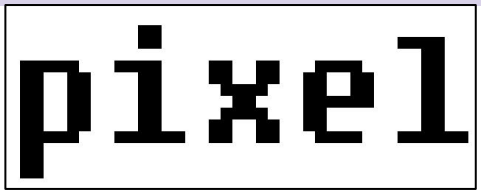

renderer process
CompositorFrame
DrawQuad
browser process
Display::DrawAndSwap
GLRenderer::DrawTileQuad
GLES2Implementation
GPU process
CompositorFrame
browser UI,
other renderers
The display compositor
puts pixels on the screen.
woohoo

review


content
<html>...</html>
renderer process
Blink
main
DOM
style
layout
compositing update
paint
impl
commit
tiling
raster
SKIA
activate
draw
OpenGL
GPU process
browser process
display

end

LIFE OF A



slides: bit.ly/lifeofapixel

feedback: skobes@chromium.org